aws

資料庫開發人員指南

Amazon Redshift

Amazon Redshift: 資料庫開發人員指南

Table of Contents

簡介

歡迎使用《Amazon Redshift 資料庫開發人員指南》。Amazon Redshift 是一種在雲端中完全受管的 PB 級資料倉儲服務。Amazon Redshift Serverless 可讓您無需佈建資料倉儲的一般組態，即可存取和分析資料。系統會自動佈建資源，並有智慧地擴展資料倉儲容量，即使是最嚴苛且無法預測的工作負載，也能為其提供快速的效能。資料倉儲閒置時不會產生費用，因此只需按實際用量支付費用。無論資料集的大小為何，您都可以立即在 Amazon Redshift 查詢編輯器 v2 或偏好的商業智慧 (BI) 工具中載入資料並開始查詢。在零管理環境中享受最優惠的價格效能和熟悉的 SQL 功能。 easy-to-use

此指南著重於使用 Amazon Redshift 來建立和管理資料倉儲。若您是以設計人員、軟體開發人員或管理員的身分在使用資料庫，此指南會提供您設計、建置、查詢和維護資料倉儲所需的資訊。

主題

必要條件

在使用本指南之前，請先閱讀 Amazon Redshift Serverless，瞭解如何完成下列任務。

- 使用 Amazon Redshift Serverless 建立資料倉儲。
- 使用 Amazon Redshift 查詢編輯器第 2 版載入範例資料
- 從 Amazon S3 載入資料。

您也應了解如何使用 SQL 用戶端，並對 SQL 語言有基本了解。

您是資料庫開發人員嗎？

如果您是第一次使用 Amazon Redshift，我們建議您閱讀 Amazon Redshift Serverless 以了解如何開始使用。

如您是資料庫使用者、資料庫設計人員、資料庫開發人員或資料庫管理員，下表有助於找出您要查詢的項目。

如果您想要...	我們建議...
了解 Amazon Redshift 資料倉儲的內部架構。	系統和架構概觀會顯示 Amazon Redshift 內部架構的高階概觀。 若您想要 Amazon Redshift Web 服務更廣泛的概觀，請前往 Amazon Redshift 產品詳細資訊頁面。
建立資料庫、資料表、使用者和其他資料庫物件。	常見的數據庫任務是 SQL 開發的基礎知識的快速介紹。 Amazon Redshift SQL 有 Amazon Redshift SQL 命令和函數，以及其他 SQL 元素的語法和範例。 設計資料表的 Amazon Redshift 最佳實務 提供了選擇排序索引鍵、散發索引鍵和壓縮編碼上建議的摘要。
了解如何設計資料表以求最佳效能。	使用自動資料表最佳化 詳述了對資料表資料欄中的資料套用壓縮，以及選擇散發索引鍵和排序索引鍵的考量事項。
載入資料。	載入資料 會說明從Amazon DynamoDB 資料表或儲存在 Amazon S3 儲存貯體中的一般檔案，載入大型資料集的程序。 載入資料的 Amazon Redshift 最佳實務 會提供快速有效地載入資料的秘訣。
管理使用者、群組和資料庫安全。	管理資料庫安全 涵蓋了資料庫安全主題。
監控系統效能並最佳化。	系統資料表和檢視參考 詳述了系統資料表和檢視，可讓您查詢資料庫的狀態，並監控查詢和程序。 另請參閱《Amazon Redshift 管理指南》https://docs.aws.amazon.com/redshift/latest/mgmt/以了解如何使用 AWS Management Console ，來檢查系統運作狀態、監控指標，以及備份和還原叢集。
分析來自非常大型資料集的資訊，並進行報告。	許多受歡迎的軟體廠商會認證 Amazon Redshift 可搭配自家產品和服務，以便讓您繼續使用現在用的工具。如需詳細資訊，請參閱 Amazon Redshift 合作夥伴頁面。 SQL 參考 有 Amazon Redshift 所支援的 SQL 運算式、命令和函數相關的所有詳細資訊。

如果您想要...	我們建議...
與 Amazon Redshift 資源和資料表互動。	請參閱《Amazon Redshift Serverless API 指南》https://docs.aws.amazon.com/redshift-serverless/latest/APIReference/Welcome.html、《Amazon Redshift API 指南》https://docs.aws.amazon.com/redshift/latest/APIReference/Welcome.html和《Amazon Redshift 資料 API 指南》https://docs.aws.amazon.com/redshift-data/latest/APIReference/Welcome.html，以進一步了解如何以程式設計方式與資源互動和執行操作。
遵循教學課程，讓您更熟悉 Amazon Redshift。	按照 Amazon Redshift 教學課程中的教學課程，進一步了解有關 Amazon Redshift 功能的資訊。

系統和架構概觀

Amazon Redshift 資料倉儲是企業級的關聯式資料庫查詢與管理系統。

Amazon Redshift 透過多種類型的應用程式來支援用戶端連線，包括商業智慧 (BI)、報告、資料與分析工具。

執行分析查詢時，會以多階段操作的方式，擷取、比較和評估大量的資料，來產生最終的結果。

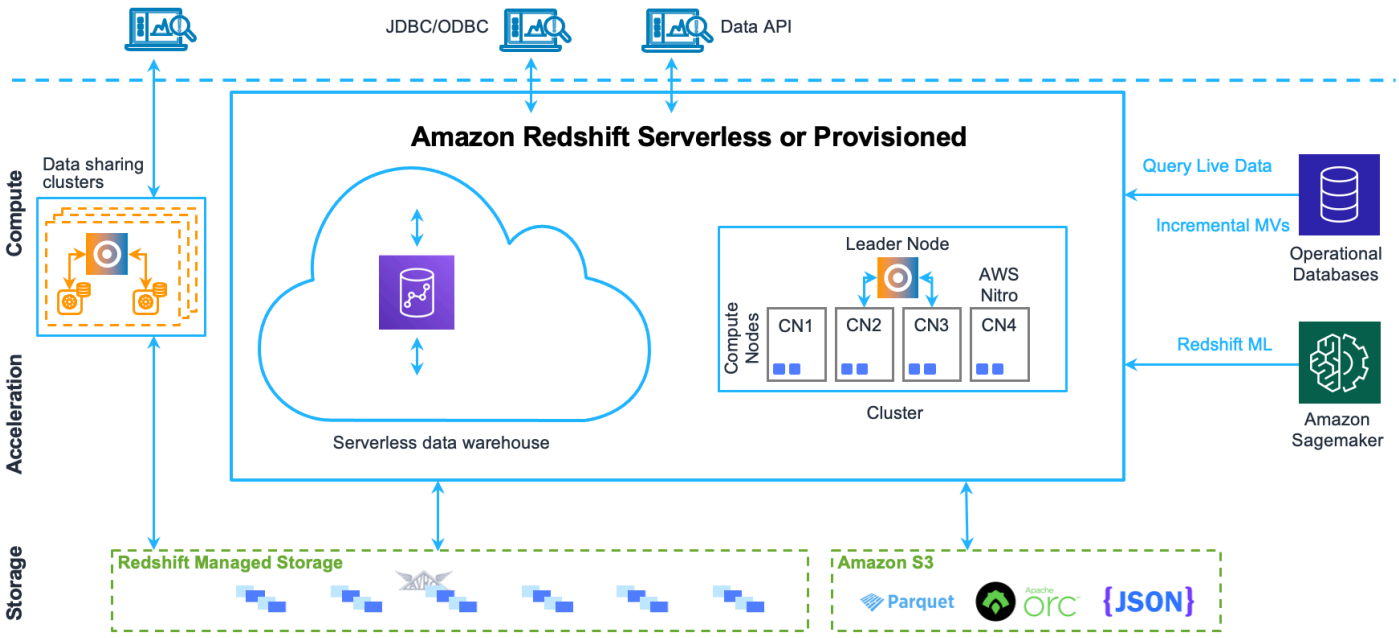
藉由結合大規模的平行處理作業、單欄式資料儲存體，和超高效率的針對式資料壓縮編碼機制，Amazon Redshift 可實現高效率的儲存與最佳化的查詢效能。本節提供 Amazon Redshift 系統架構的介紹。

主題

- 資料倉儲系統架構
- 效能
- 單欄式儲存
- 工作負載管理
- 搭配其他服務使用 Amazon Redshift

資料倉儲系統架構

本節介紹 Amazon Redshift 資料倉儲架構的元素，如下圖所示。



用戶端應用程式

Amazon Redshift 整合了各種資料載入和 ETL (擷取、轉換和載入) 工具，以及商業智慧 (BI) 報告、資料探勘和分析工具。Amazon Redshift 是以開放式標準 PostgreSQL 為基礎，因此大多數現有的 SQL 用戶端應用程式都只需進行最少的變更即可運作。關於 Amazon Redshift SQL 與 PostgreSQL 之間重要差異的相關資訊，請參閱 Amazon Redshift 和 PostgreSQL。

叢集

Amazon Redshift 資料倉儲的核心基礎設施元件是叢集。

叢集是由一或多個運算節點所組成。如果為叢集佈建了兩個或多個運算節點，則會有另外的領導節點負責統籌運算節點和處理外部通訊。您的用戶端應用程式只會直接和領導節點互動，外部應用程式不會知道運算節點的存在。

領導節點

領導節點會管理與用戶端程式的通訊，以及和運算節點的所有通訊。此節點會剖析和制定執行計畫，來執行資料庫的操作，尤其是取得複雜查詢結果所需的一系列步驟。領導節點會根據執行計畫，來編譯程式碼、將編譯過的程式碼分發到運算節點，並指派資料的各部分給每個運算節點。

只有當查詢參考儲存於運算節點上的資料表時，領導節點才會將 SQL 陳述式分發到運算節點。所有其他查詢僅在領導節點上執行。Amazon Redshift 的設計目的是僅在領導節點上實作某些 SQL 函數。如果查詢使用這些函式的任一個，而且參考位於運算節點上的資料表，則將會傳回錯誤。如需詳細資訊，請參閱 領導節點上所支援的 SQL 函數。

運算節點

領導節點會針對執行計畫的個別元素，來編譯程式碼，並將程式碼指派給個別的運算節點。運算節點會執行編譯過的程式碼，並將中間的結果傳回領導節點，以進行最終的彙總。

每個運算節點都具有自己專屬的 CPU 和記憶體 (根據節點類型決定)。隨著工作負載的增長，您可以藉由增加節點的數量、將節點的類型升級，或是同時實行這兩種做法，來增加叢集的運算容量。

Amazon Redshift 會針對您的運算需求，提供數種節點類型。如需每個節點類型的詳細資訊，請參閱《Amazon Redshift 管理指南》中的 < Amazon Redshift 叢集 > 。

Redshift 受管儲存

資料倉儲資料會儲存在個別的儲存層 Redshift 受管儲存 (RMS) 中。RMS 提供使用 Amazon S3 儲存體將儲存空間擴展到 PB 級的能力。RMS 可讓您單獨完成運算和儲存體的擴展與付費，因此您可以只根據運算需求調整叢集的大小。它會自動使用高效能 SSD 型本機儲存做為第 1 層快取。它還會利用資料區塊溫度、資料區塊存留時間和工作負載模式等最佳化功能提供高效能，同時在需要時自動將儲存擴展到 Amazon S3，而無需採取任何動作。

節點配量

運算節點可分割為分割。每個分割會分配到節點的一部分記憶體和磁碟空間，並且用以處理指派給節點的部分工作負載。領導節點會負責將資料分發給分割，並將任何查詢或其他資料庫操作的工作負載，分派給這些分割。這些分割接著會平行運作，以完成操作。

每個節點的分割數目，取決於叢集的節點大小。如需每個節點大小有多少配量的相關資訊，請移至《Amazon Redshift 管理指南》中的 < 關於叢集和節點 > 。

建立資料表時，您可以選擇性地指定一個欄做為分佈索引鍵。當資料載入資料表時，會根據為資料表定義的分佈索引鍵，將資料列分發給節點分割。選擇理想的分佈索引鍵，可讓 Amazon Redshift 利用平行處理作業，以高效率的方式載入資料和執行查詢。如需關於選擇分佈索引鍵的詳細資訊，請參閱選擇最佳的分佈方式。

內部網路

Amazon Redshift 善用了高頻寬連線、鄰近位置與自訂通訊協定，在領導節點和運算節點之間，提供私密的超高速網路通訊。運算節點會在獨立的隔離網路中執行，用戶端應用程式絕對不會直接存取此網路。

資料庫

叢集包含一個或多個資料庫。使用者資料儲存於運算節點上。您的 SQL 用戶端會和領導節點進行通訊，然後再由領導節點統籌使用運算節點的查詢執行作業。

Amazon Redshift 是關聯式資料庫管理系統 (RDBMS)，因此和其他的 RDBMS 應用程式相容。雖然其提供與典型 RDBMS 相同的功能 (包括線上交易處理 (OLTP) 功能，例如插入和刪除資料)，不過 Amazon Redshift 最適合應用於超大資料集的高效能分析與報告。

Amazon Redshift 是以 PostgreSQL 為基礎。在設計和開發您的資料倉儲應用程式時，您需要考量 Amazon Redshift 與 PostgreSQL 之間的許多很重要的差異。如需 Amazon Redshift SQL 與 PostgreSQL 之間差異的詳細資訊，請參閱 Amazon Redshift 和 PostgreSQL。

效能

Amazon Redshift 運用這些效能功能，實現超快速的查詢執行。

主題

- 大規模平行處理
- 單欄式資料儲存體
- 資料壓縮
- 查詢最佳化工具
- 結果快取
- 經過編譯的程式碼

大規模平行處理

大規模平行處理 (MPP) 作業讓處理大量資料的最複雜查詢，也能快速地執行。多重運算節點進行所有查詢處理作業，最後再將結果彙總 (每個節點的核心會針對整個資料的一部分，執行相同的編譯後查詢片段)。

Amazon Redshift 會將資料表的列分發給運算節點，以平行處理資料。透過為每個資料表選擇適合的分佈索引鍵，您可以將資料的分發最佳化，進而平衡工作負載，並且將資料在節點之間的移動次數減到最少。如需詳細資訊，請參閱 選擇最佳的分佈方式。

從一般檔案載入資料，可透過將工作負載分散到多個節點，一邊同時從多個檔案讀取，來發揮平行處理的效益。如需如何將資料載入資料表的相關資訊，請參閱載入資料的 Amazon Redshift 最佳實務。

單欄式資料儲存體

資料庫資料表的單欄式儲存方式，大幅地降低了整體磁碟輸入/輸出需求，成為實現最佳化分析查詢效能的重要因素。以單欄式儲存資料庫資料表的資訊，可減少磁碟輸入/輸出要求的數量，和需要從磁碟

載入的資料量。載入記憶體的資料量減少，即可讓 Amazon Redshift 在執行查詢時，進行更多的記憶體內處理作業。如需更詳細的說明，請參閱單欄式儲存。

以適當地方式將欄排序時，查詢處理器就能夠快速地篩選掉龐大的資料區塊子集。如需詳細資訊，請參閱 選擇最佳的排序索引鍵。

資料壓縮

資料壓縮可降低對儲存空間的需求，進而減少磁碟輸入/輸出，提升查詢作業的效能。執行查詢作業時，會將壓縮的資料讀入記憶體，然後在進行查詢時將資料解壓縮。載入記憶體的資料量減少，即可讓 Amazon Redshift 配置更多的記憶體來分析資料。由於單欄式儲存會依順序儲存類似的資料，Amazon Redshift 就能夠採用與單欄式資料類型特別相關的適應性壓縮編碼機制。對資料表欄進行資料壓縮的最佳方式，就是在載入資料表和資料時，讓 Amazon Redshift 套用最佳化的壓縮編碼機制。如要進一步了解自動資料壓縮，請參閱利用自動壓縮載入資料表。

查詢最佳化工具

Amazon Redshift 查詢執行引擎採用支援 MPP 的查詢最佳化工具，也運用了欄式導向的資料儲存體。Amazon Redshift 查詢最佳化工具建置了重大的增強與擴充功能，可用來處理複雜的分析查詢作業，這類查詢經常包含多重資料表的合併、子查詢和彙總操作。若要進一步了解如何將查詢作業最佳化，請參閱調校查詢效能。

結果快取

為了縮短查詢執行期並改善系統效能，Amazon Redshift 會將特定查詢類型的結果快取在領導者節點的記憶體中。當使用者提交查詢時，Amazon Redshift 會針對結果快取，檢查是否建立了查詢結果的有效快取副本。如果在結果快取中找到符合者，Amazon Redshift 會使用快取的結果，不會執行查詢。使用者並不會察覺到結果快取作業的進行。

結果快取預設為開啟。若要關閉目前工作階段的結果快取作業，請將 enable_result_cache_for_session 參數設定為 `off`。

當下列所有情況都符合時，Amazon Redshift 會使用新查詢的快取結果：

- 提交查詢的使用者，對查詢中所使用的物件具有存取許可。
- 查詢中的資料表或檢視尚未經過修改。
- 查詢並未使用每次執行時都必須求值的函式，例如 GETDATE。
- 查詢並未參考 Amazon Redshift Spectrum 外部資料表。

- 可能影響查詢結果的組態參數不變。
- 查詢在語法上符合快取的查詢。

為了最大限度地提高快取效率並有效利用資源，Amazon Redshift 不會快取某些大型查詢結果集。Amazon Redshift 會根據許多因素判斷是否要快取查詢結果。這些因素包括快取中的項目數量，以及 Amazon Redshift 叢集的執行個體類型。

若要判斷查詢作業是否使用了結果快取，請查詢 SVL_QLOG 系統畫面。如果查詢使用了結果快取，source_query 欄會傳回來源查詢的查詢 ID。如果未使用結果快取，source_query 欄的值為 NULL。

下列範例顯示，由 userid 104 和 userid 102 所提交的查詢，使用了 userid 100 所執行查詢的結果快取。

```
select userid, query, elapsed, source_query from svl_qlog
where userid > 1
order by query desc;

userid | query  | elapsed  | source_query
-------+--------+----------+-------------
   104 | 629035 |       27 |       628919
   104 | 629034 |       60 |       628900
   104 | 629033 |       23 |       628891
   102 | 629017 |  1229393 |
   102 | 628942 |       28 |       628919
   102 | 628941 |       57 |       628900
   102 | 628940 |       26 |       628891
   100 | 628919 | 84295686 |
   100 | 628900 | 87015637 |
   100 | 628891 | 58808694 |
```

經過編譯的程式碼

領導節點會將經過完整最佳化編譯的程式碼，分發給叢集的所有節點。編譯查詢可省去與直譯器相關的資源耗用，進而提升執行期速度，尤其是複雜的查詢作業。編譯後的程式碼會經過快取，並在同一個叢集上的工作階段之間共用。因此，未來執行相同查詢的速度會更快，即使使用不同的參數也應是如此。

查詢執行引擎會針對 JDBC 和 ODBC 連線通訊協定編譯不同的程式碼，如此，使用不同通訊協定的兩個用戶端，各會產生編譯程式碼的首次成本。使用相同通訊協定的用戶端，可從共用快取的程式碼而受惠。

單欄式儲存

資料庫資料表的單欄式儲存方式是實現最佳化分析查詢效能的重要因素，因為其大幅地降低了整體磁碟輸入/輸出需求。這麼做可降低您需要從磁碟載入的資料量。

下列的一系列圖解，說明單欄式資料儲存如何實現效率，進而提升將資料擷取到記憶體時的效率。

第一張圖顯示資料庫資料表的記錄，通常是如何以列的形式儲存到磁碟區塊中。

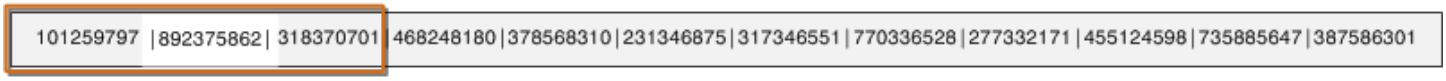
SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

Block 1 Block 2 Block 3

在典型的關聯式資料庫資料表中，每列會包含單一筆記錄的欄位值。在列式資料庫儲存法中，資料區塊會依順序儲存每個連續資料欄的值，這構成了一個整列。如果區塊的容量小於記錄的大小，則整筆記錄所佔的儲存空間可能會超過一個區塊。如果區塊的容量大於記錄的大小，則整筆記錄所佔的儲存空間可能會少於一個區塊，造成磁碟空間使用上的無效率。在線上交易處理 (OLTP) 應用中，大部分的異動作業牽涉到經常性的讀取和寫入整個記錄的所有值 (一次通常讀取和寫入一筆記錄或少量的幾筆記錄)。因此，OLTP 資料庫採用列式儲存法是最理想的。

下一張圖解顯示，使用單欄式儲存法時，每欄的值如何按順序儲存到磁碟區塊中。

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL



Block 1

使用單欄式儲存法時，每個資料區塊會儲存多個資料列單欄的值。當記錄進到系統時，Amazon Redshift 會在不被察覺的狀況下，將各欄的資料轉換為單欄式儲存模式。

在這個簡化的範例中，相較於列式儲存法，使用單欄式儲存法時，每個資料區塊可儲存多達三倍的記錄欄位值。這代表相較於列式儲存法，針對相同數目的記錄讀取相同數量的欄位值時，只需要進行三分之一的輸入/輸出操作。實際上，所使用的資料表如果包含龐大數量的欄數和列數時，單欄式儲存法的儲存效率甚至會更高。

另一個額外的優點是，由於每個區塊都儲存相同類型的資料，因此區塊資料可以採用專為欄資料類型而選擇的壓縮方法，以進一步減少磁碟空間和 I/O。如需根據資料類型決定壓縮編碼方法的詳細資訊，請參閱 壓縮編碼。

在磁碟上儲存資料所省下的空間，也可沿用到擷取資料然後將資料儲存到記憶體。由於許多的資料庫操作一次只需存取或操作一個或少量的欄，因此藉由只擷取查詢實際所需的資料欄區塊，可以節省記憶體的空間。OLTP 交易通常會處理少量記錄資料列的大部分欄或所有欄，資料倉儲查詢則通常只會讀取龐大數量資料列的幾個欄。這代表針對相同數目的資料列讀取相同數量的欄位值時，只需要進行部分的輸入/輸出操作。其使用處理資料列區塊時所需的一小部分記憶體。實際上，所使用的資料表如果包含龐大數量的欄數和列數時，欄式儲存法的儲存效率會按比例提高。例如，假設資料表包含 100 個資料欄。使用 5 個資料欄的查詢作業，將只需要讀取資料表中所包含資料的 5%。大型資料庫可能會有數十億或甚至數兆的記錄，都能重複實現這種節省效率。相較之下，列式資料庫也會讀取包含其他 95 個非必要資料欄的資料區塊。

典型的資料庫區塊大小範圍介於 2 KB 到 32 KB 之間。Amazon Redshift 使用效率更高的 1 MB 區塊大小，這在執行任何的資料庫載入作業，或查詢作業執行所包含的其他操作時，可進一步減少所需的 I/O 請求數量。

工作負載管理

Amazon Redshift 工作負載管理 (WLM) 功能可讓使用者彈性地管理工作負載內的優先順序，如此，可快速執行完成的簡短查詢作業，就不會因為排在需要長時間執行的查詢之後，而在佇列中塞車。

Amazon Redshift WLM 會在執行期根據服務類別來建立查詢佇列，服務類別定義各種佇列的組態參數，包括內部系統佇列和使用者可存取的佇列。從使用者的角度來看，使用者可存取的服務類別和佇列在功能上是相同的。為了保持一致，本文件使用佇列一詞，來同時表示使用者可存取的服務類別，以及執行時間佇列。

當您執行查詢時，WLM 會根據使用者的使用者群組，或是藉由比對查詢組態中所列的查詢群組，和使用者在執行時間所設定的查詢群組標籤，來指派查詢作業給佇列。

目前，使用預設參數群組的叢集依預設是使用自動 WLM。自動 WLM 可管理查詢並行和記憶體配置。如需詳細資訊，請參閱 實作自動 WLM。

Amazon Redshift 會使用手動 WLM，設定一個並行層級為 5 的佇列 (此佇列可讓最多 5 個查詢同時執行)，加上一個預先定義、並行層級為 1 的超級使用者佇列。您最多可以設定 8 個佇列。每個佇列最多可設定 50 的並行層級。所有使用者定義佇列 (不包括進階使用者佇列) 的總計並行層級上限為 50。

修改 WLM 組態最簡單的方式是使用 Amazon Redshift 管理主控台。您也可以使用 Amazon Redshift 命令列介面 (CLI) 或 Amazon Redshift API。

如需建置和使用工作負載管理機制的相關資訊，請參閱實作工作負載管理。

搭配其他服務使用 Amazon Redshift

Amazon Redshift 與其他 AWS 服務整合，讓您能夠使用資料安全功能快速、可靠地移動、轉換和載入資料。

在 Amazon Redshift 和 Amazon S3 之間移動資料

Amazon Simple Storage Service (Amazon S3) 是將資料儲存在雲端中的 Web 服務。Amazon Redshift 利用平行處理功能，從儲存在 Amazon S3 儲存貯體中的多個資料檔案讀取和載入資料。如需詳細資訊，請參閱 從 Amazon S3 載入資料。

您也可以利用平行處理模式，從 Amazon Redshift 資料倉儲將資料匯出至 Amazon S3 上的多個資料檔案。如需詳細資訊，請參閱 卸載資料。

搭配 Amazon DynamoDB 使用 Amazon Redshift

Amazon DynamoDB 是全受管的 NoSQL 資料庫服務。您可以使用 COPY 命令，從單一 Amazon DynamoDB 資料表載入具有資料的 Amazon Redshift 資料表。如需詳細資訊，請參閱 從 Amazon DynamoDB 資料表載入資料。

透過 SSH 從遠端主機匯入資料

您可以在 Amazon Redshift 中使用 COPY 指令，從一個或多個遠端主機載入資料，例如 Amazon EMR 叢集、Amazon EC2 執行個體或其他電腦。COPY 會使用 SSH 連接至遠端主機，然後在遠端主機執行命令來產生資料。Amazon Redshift 支援多個同時連線。COPY 指令會以平行方式，從多個主機資源讀取和載入輸出的資料。如需詳細資訊，請參閱 從遠端主機載入資料。

使用自動化資料載入 AWS Data Pipeline

您可以使 AWS Data Pipeline 用自動化資料移入和轉出 Amazon Redshift。透過使用的內建排程功能 AWS Data Pipeline，您可以排程和執行週期性工作，而不必撰寫自己的複雜資料傳輸或轉換邏輯。

例如，您可以設定重複性的任務，自動將資料從 Amazon DynamoDB 複製到 Amazon Redshift。如需將資料從 Amazon S3 定期移動到 Amazon Redshift 的建立管道程序的教學課程，請參閱 AWS Data Pipeline 開發人員指南 AWS Data Pipeline中的使用方法將資料複製到 Amazon Redshift。

使用 AWS Database Migration Service (AWS DMS) 移轉資料

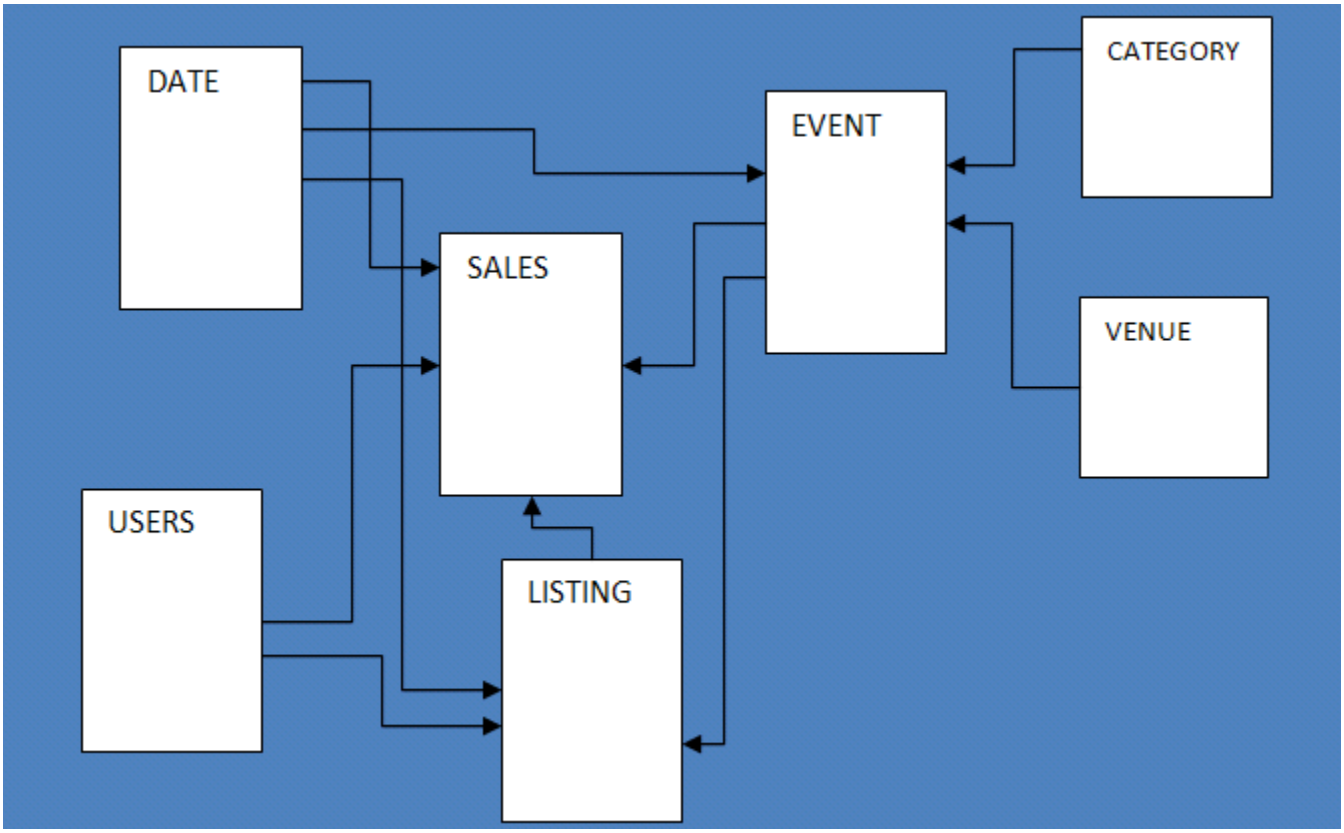
您可以使 AWS Database Migration Service用將數據遷移到 Amazon Redshift。 AWS DMS 可以在最廣泛使用的商業和開放原始碼資料庫 (例如甲骨文、PostgreSQL、Microsoft SQL 伺服器、Amazon Redshift、Aurora 資料庫叢集、DynamoDB 資料庫、Amazon S3、MariaDB 和 MySQL) 之間遷移資料。如需詳細資訊，請參閱使用 Amazon Redshift 資料庫做為 AWS Database Migration Service的目標。

範本資料庫

主題

- CATEGORY 資料表
- DATE 資料表
- EVENT 資料表
- VENUE 資料表
- USERS 資料表
- LISTING 資料表
- SALES 資料表

Amazon Redshift 文件中的多數範例都會使用名為 TICKIT 的範本資料庫。此小型資料庫包含七個資料表：兩個事實資料表和五個維度。您可以按照亞馬遜紅移入門指南中的步驟 4：將資料從 Amazon S3 載入到亞馬 Amazon Redshift 中的步驟中的步驟載入 TICKIT 資料集。



此範例資料庫應用程式可協助對虛擬 TICKIT 網站的追蹤銷售活動之分析，使用者會在該網站中線上購買和銷售運動活動、表演和演唱會的門票。分析可特別辨識門票隨著時間的動向、賣方的成功比率以及熱賣活動、會場和季節。分析可使用此資訊來為經常造訪網站的買方和賣方提供誘因，以吸引新使用者和促使廣告和促銷。

例如，下列查詢會根據在 2008 所售的門票數，找到在聖地牙哥中前五大賣方：

```
select sellerid, username, (firstname ||' '|| lastname) as name,
city, sum(qtysold)
from sales, date, users
where sales.sellerid = users.userid
and sales.dateid = date.dateid
and year = 2008
and city = 'San Diego'
group by sellerid, username, name, city
order by 5 desc
limit 5;

sellerid | username |       name        |   city    | sum
----------+----------+-------------------+-----------+-----
49977 | JJK84WTE | Julie Hanson      | San Diego |  22
```

```
19750 | AAS23BDR | Charity Zimmerman | San Diego |  21
29069 | SVL81MEQ | Axel Grant        | San Diego |  17
43632 | VAG08HKW | Griffin Dodson    | San Diego |  16
36712 | RXT40MKU | Hiram Turner      | San Diego |  14
(5 rows)
```

在指南中用於此範例的資料庫包含小型資料集，兩個事實資料表各包含少於 200,000 個的列，在 CATEGORY 資料表中來自 11 個列的維度範圍高達約 50,000 列 (在 USERS 資料表)。

此指南中的資料庫範例特別示範 Amazon Redshift 資料表設計的主要功能：

- 資料分佈
- 資料排序
- 直欄式壓縮

CATEGORY 資料表

欄名稱	資料類型	描述
CATID	SMALLINT	主要索引鍵，每個列的唯一 ID 值。每個列代表購買或賣出之特定類型的活動門票。
CATGROUP	VARCHAR(10)	活動群組的描述性名稱，例如 **Shows** 和 **Sports**。
CATNAME	VARCHAR(10)	群組中活動類型的簡短描述性名稱，例如 **Opera** 和 **Musicals**。
CATDESC	VARCHAR(50)	活動類型的較長描述性名稱，例如 **Musical theatre**。

DATE 資料表

欄名稱	資料類型	描述
DATEID	SMALLINT	主要索引鍵，每個列的唯一 ID 值。每個列代表日曆年中的一天。
CALDATE	DATE	日曆日期，例如 **2008-06-24** 。

欄名稱	資料類型	描述
DAY	CHAR(3)	一週中的一天 (短格式)，例如 **SA**。
WEEK	SMALLINT	週編號，例如 **26**。
MONTH	CHAR(5)	月名稱 (短格式)，例如 **JUN**。
QTR	CHAR(5)	季編號 (**1** 到 **4**)。
YEAR	SMALLINT	四位數年份 (**2008**)。
HOLIDAY	BOOLEAN	標記表示該日是否為國定假日 (美國)。

EVENT 資料表

欄名稱	資料類型	描述
EVENTID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每個列代表個別的項目，這些活動會在特定時間在特定的會場舉辦。
VENUEID	SMALLINT	外部索引鍵會參考 VENUE 資料表。
CATID	SMALLINT	外部索引鍵會參考 CATEGORY 資料表。
DATEID	SMALLINT	外部索引鍵會參考 DATE 資料表。
EVENTNAME	VARCHAR(200)	活動名稱，例如 **Hamlet** 或 **La Traviata**。
STARTTIME	TIMESTAMP	活動的完整日期和開始時間，例如 **2008-10-10 19:30:00**。

VENUE 資料表

欄名稱	資料類型	描述
VENUEID	SMALLINT	主要索引鍵，每個列的唯一 ID 值。每個列代表活動舉辦的特定會場。
VENUENAME	VARCHAR(100)	場所的確切名稱，例如 **Cleveland Browns Stadium**。
VENUECITY	VARCHAR(30)	城市名稱，例如 **Cleveland** 。
VENUESTATE	CHAR(2)	兩個字母的州或省縮寫 (美國和加拿大)，例如 **OH**。
VENUESEATS	INTEGER	會場可用座位數上限 (如果知道)，例如 **73200**。為達示範目的，此欄位包含一些空值和零。

USERS 資料表

欄名稱	資料類型	描述
USERID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每列代表註冊的使用者 (買方或賣方或兩者)，這些使用者已列出或購買至少一個活動的門票。
USERNAME	CHAR(8)	8 字元英數使用者名稱，例如 **PGL08LJI**。
FIRSTNAME	VARCHAR(30)	使用者的名字，例如 **Victor**。
LASTNAME	VARCHAR(30)	使用者的姓氏，例如 **Hernandez** 。
CITY	VARCHAR(30)	使用者的住家所在城市，例如 **Naperville** 。
STATE	CHAR(2)	使用者的住家所在州，例如 **GA**。
EMAIL	VARCHAR(100)	使用者的電子郵件地址，此欄位包含隨機 Latin 值，例如 **turpis@accumsanlaoreet.org** 。

欄名稱	資料類型	描述
PHONE	CHAR(14)	使用者的 14 個字元的電話號碼，例如 **(818) 765-4255**。
LIKESPORTS, ...	BOOLEAN	一系列 10 個不同欄位，您可透過 **true** 和 **false** 值識別使用者的喜好。

LISTING 資料表

欄名稱	資料類型	描述
LISTID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每個列代表一組特定活動之門票的清單。
SELLERID	INTEGER	外部索引鍵會參考 USERS 資料表，識別正在銷售門票的使用者。
EVENTID	INTEGER	外部索引鍵會參考 EVENT 資料表。
DATEID	SMALLINT	外部索引鍵會參考 DATE 資料表。
NUMTICKETS	SMALLINT	可供銷售的門票數，例如 **2** 或 **20**。
PRICEPERTICKET	DECIMAL(8,2)	個別門票的定價，例如 **27.00** 或 **206.00**。
TOTALPRICE	DECIMAL(8,2)	此傳單的總價 (NUMTICKETS*PRICEPERTICKET)。
LISTTIME	TIMESTAMP	張貼此傳單的完整日期和時間，例如 **2008-03-18 07:19:35**。

SALES 資料表

欄名稱	資料類型	描述
SALESID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每個列代表特定活動一或多個門票的銷售 (如特定清單中所提供)。
LISTID	INTEGER	外部索引鍵會參考 LISTING 資料表。
SELLERID	INTEGER	外部索引鍵會參考 USERS 資料表 (銷售門票的使用者)。
BUYERID	INTEGER	外部索引鍵會參考 USERS 資料表 (購買門票的使用者)。
EVENTID	INTEGER	外部索引鍵會參考 EVENT 資料表。
DATEID	SMALLINT	外部索引鍵會參考 DATE 資料表。
QTYSOLD	SMALLINT	門票銷售數，從 **1** 到 **8**。(您最多可以在單一交易中銷售 8 張門票。)
PRICEPAID	DECIMAL(8,2)	為門票支付的總價，例如 **75.00** 或 **488.00**。門票的個別價格為 PRICEPAID/QTYSOLD。
COMMISSION	DECIMAL(8,2)	公司會從銷售中收取 15% 佣金，例如 **11.25** 或 **73.20**。賣方可獲得 PRICEPAID 值的 85%。
SALETIME	TIMESTAMP	銷售結束的完整日期和時間，例如 **2008-05-24 06:21:47**。

Amazon Redshift 最佳實務

接下來，您可以找到有關規劃概念驗證、設計資料表、載入資料至資料表，以及為 Amazon Redshift 撰寫查詢的最佳實務，以及有關使用 Amazon Redshift Advisor 的討論。

Amazon Redshift 不同於其他 SQL 資料庫系統。為了完全實現 Amazon Redshift 架構的優點，您必須特別設計、建置及載入您的資料表，以使用大規模平行處理、單欄式資料儲存體，以及單欄式資料壓縮。如果您的資料載入與查詢執行時間比預期長，或比您希望的長，表示您可能忽略了關鍵資訊。

如果您是有經驗的 SQL 資料庫開發人員，在開始開發您的 Amazon Redshift 資料倉儲之前，強烈建議您檢閱本主題。

如果您是開發 SQL 資料庫的新手，本主題並非最佳的起點。我們建議您先閱讀一般資料庫工作，然後自行嘗試範例。

在本主題中，您可以找到最重要的開發原則概觀，以及實作這些原則的具體技巧、範例及最佳實務。沒有任何單一實務可以套用至每個應用程式。在完成資料庫設計之前，請評估所有的選項。如需詳細資訊，請參閱 使用自動資料表最佳化、載入資料、調校查詢效能，以及參考章節。

主題

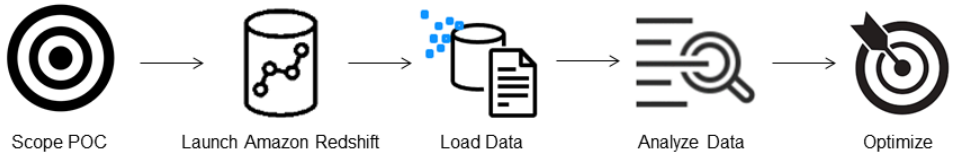
- 為 Amazon Redshift 進行概念證明（POC）
- 設計資料表的 Amazon Redshift 最佳實務
- 載入資料的 Amazon Redshift 最佳實務
- 設計查詢的 Amazon Redshift 最佳實務
- 使用 Amazon Redshift Advisor 的建議

為 Amazon Redshift 進行概念證明（POC）

Amazon Redshift 是一種熱門的雲端資料倉儲，提供全受管雲端服務，可與組織的 Amazon 簡易儲存服務資料湖、即時串流、機器學習 (ML) 工作流程、交易工作流程等整合。以下各節將引導您完成在 Amazon Redshift 上進行概念驗證 (POC) 的過程。此處的資訊可協助您為 POC 設定目標，並利用可自動佈建和設定 POC 服務的工具。

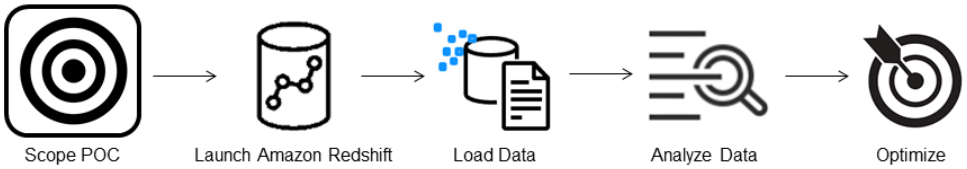
> **Note**
>
> 如需以 PDF 格式提供此資訊的副本，請在亞馬遜紅移資源頁面上選擇執行您自己的 Redshift POC 連結。

執行 Amazon Redshift 的 POC 時，您可以測試、證明和採用各種功能，包括 best-in-class 安全功能、彈性擴展、輕鬆整合和擷取，以及靈活的分散式資料架構選項。



請按照以下步驟進行成功的 POC。

步驟 1：設定 POC 的範圍



執行 POC 時，您可以選擇使用自己的資料，也可以選擇使用基準測試資料集。當您選擇自己的資料時，您可以針對資料執行自己的查詢。使用基準測試資料，範例查詢會隨基準提供。如果您還沒準備好使用自己的資料進行 POC，請參閱使用範例資料集以取得更多詳細資訊。

一般而言，我們建議您將兩週的資料用於 Amazon Redshift POC。

通過執行以下操作開始：

1. 確定您的業務和功能需求，然後向後工作。常見的範例包括：更快的效能、更低的成本、測試新的工作負載或功能，或 Amazon Redshift 與其他資料倉儲之間的比較。
2. 設定成為 POC 成功標準的特定目標。例如，通過更快的性能，列出您希望加速的前五個進程的列表，並包括當前的運行時間以及所需的運行時間。這些可以是報告、查詢、ETL 程序、資料擷取或您目前的痛點。

3. 確定運行測試所需的特定範圍和工件。您需要哪些資料集才能遷移或持續導入 Amazon Redshift，以及執行測試以根據成功標準進行測量需要哪些查詢和程序？ 有兩種方式可以進行：

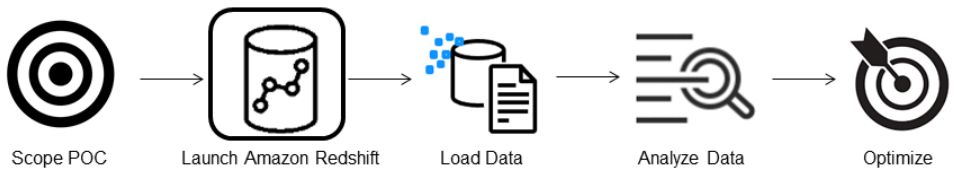
   攜帶您自己的資料

   - 要測試您自己的數據，請提出測試成功標準所需的最小可行數據成品列表。例如，如果您目前的資料倉儲有 200 個資料表，但您要測試的報表只需要 20 個資料表，則只需使用較小的資料表子集，就可以更快地執行 POC。

   使用範例資料集

   - 如果您沒有準備好自己的資料集，您仍然可以使用業界標準的基準資料集 (例如 TPC-DS 或 TPC-H) 開始在 Amazon Redshift 上執行 POC，並執行範例基準測試查詢來利用 Amazon Redshift 的強大功能。您可以在建立 Amazon Redshift 資料倉儲之後，從您的資料倉儲中存取這些資料集。如需如何存取這些資料集和範例查詢的詳細指示，請參閱步驟 2：啟動 Amazon Redshift。

步驟 2：啟動 Amazon Redshift



Amazon Redshift 透過快速、簡單且安全的大規模雲端資料倉儲，加快您獲得洞見的時間。您可以在 Redshift 無伺服器主控台上啟動倉儲，並在幾秒鐘內從資料獲得深入解析，以快速開始。使用 Redshift 無伺服器，您可以專注於交付業務成果，而不必擔心管理資料倉儲的問題。

設定 Amazon Redshift 無伺服器

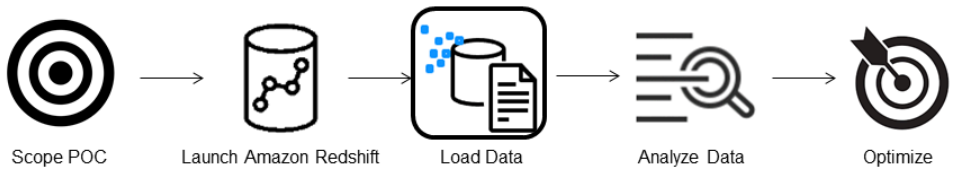
第一次使用 Redshift 無伺服器時，主控台會引導您完成啟動倉儲所需的步驟。您也可能有資格獲得帳戶中 Redshift 無伺服器用量的抵免額度。如需有關選擇免費試用的詳細資訊，請參閱 Amazon Redshift 免費試用版。遵循 Amazon Redshift Redshift 入門指南中使用 Redshift 無伺服器建立資料倉儲中的步驟，以使用 Redshift 無伺服器建立資料倉儲。如果您沒有想要載入的資料集，本指南也會提供如何載入範例資料集的步驟。

如果您之前已在帳戶中啟動 Redshift 無伺服器，請按照 Amazon Redshift 管理指南中的命名空間建立工作群組中的步驟進行操作。倉儲可用之後，您可以選擇載入 Amazon Redshift 中提供的範例資料。

如需使用 Amazon Redshift 查詢編輯器 v2 載入資料的相關資訊，請參閱 Amazon Redshift 管理指南中的載入範例資料。

如果您要使用自己的資料而不是載入範例資料集，請參閱步驟 3：載入資料。

步驟 3：載入資料



啟動 Redshift 無伺服器之後，下一步就是載入 POC 的資料。無論您要上傳簡單的 CSV 檔案、從 S3 擷取半結構化資料，還是直接串流資料，Amazon Redshift 都能提供彈性，讓您能夠快速輕鬆地將資料從來源移至 Amazon Redshift 表格。

選擇下列其中一種方法來載入資料。

上傳本機檔案

若要快速擷取和分析，您可以使用 Amazon Redshift 查詢編輯器 v2 輕鬆地從本機桌面載入資料檔案。它具有處理各種格式的文件，例如 CSV，JSON，AVRO，實木複合地板，ORC 等的能力。若要讓身為管理員的使用者能夠使用查詢編輯器 v2 從本機桌面載入資料，您必須指定通用的 Amazon S3 儲存貯體，而且使用者帳戶必須設定適當的許可。您可以使用查詢編輯器 V2 在 Amazon Redshift 中輕鬆安全地追蹤資料載入以取得 step-by-step 指導。

加載 Amazon S3 文件

若要將資料從 Amazon S3 儲存貯體載入 Amazon Redshift，請先使用 COPY 命令，指定來源 Amazon S3 位置，並將 Amazon Redshift 表作為目標。確保已正確設定 IAM 角色和許可，以允許 Amazon Redshift 存取指定的 Amazon S3 儲存貯體。遵循教學課程：從 Amazon S3 載入資料以取得 step-by-step 指導。您也可以在查詢編輯器 v2 中選擇「載入資料」選項，直接從 S3 儲存貯體載入資料。

持續資料擷取

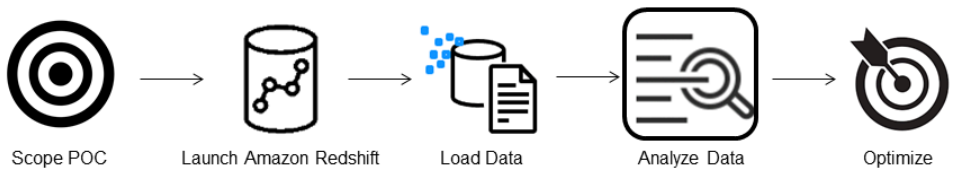
自動複製 (處於預覽狀態) 是 COPY 命令的延伸，可自動從 Amazon S3 儲存貯體連續載入資料。當您建立複製任務時，Amazon Redshift 會偵測在指定路徑中建立新的 Amazon S3 檔案時，然後自動載入這些檔案，而無需您介入。Amazon Redshift 會追蹤載入的檔案，以確認檔案只載入一次。如需如何建立複製工作的指示，請參閱 COPY JOB (預覽)

> **Note**
>
> 自動複製目前處於預覽狀態，僅在特定 AWS 區域的佈建叢集中受支援。若要建立自動複製的預覽叢集，請參閱持續從 Amazon S3 擷取檔案 (預覽)。

載入串流資料

串流擷取提供低延遲、高速的串流資料擷取功能，從 Amazon Kinesis Data Streams 和 Amazon Managed Streaming for Apache Kafka 匯入 Amazon Redshift。Amazon Redshift 串流擷取使用具體化視觀表，該視觀表會使用 auto 重新整理直接從串流更新。具體化視觀表會對應至串流資料來源。您可以在串流資料上執行篩選和彙總，做為具體化視觀表定義的一部分。如需從串流載入資料的 step-by-step 指引，請參閱 Amazon Kinesis Data Streams 入門或適用於 Apache Kafka 的 Amazon 受管串流入門。

步驟 4：分析您的資料



在建立 Redshift 無伺服器工作群組和命名空間並載入資料之後，您可以從 Redshift 無伺服器主控台的導覽面板開啟查詢編輯器 v2，立即執行查詢。您可以使用查詢編輯器 v2，針對自己的資料集測試查詢功能或查詢效能。

使用 Amazon Redshift 查詢編輯器 v2 進行查詢

您可以從 Amazon Redshift 主控台存取查詢編輯器 v2。請參閱使用 Amazon Redshift 查詢編輯器 v2 簡化資料分析，以取得如何使用查詢編輯器 v2 設定、連接和執行查詢的完整指南。

或者，如果你想運行一個負載測試作為你的 POC 的一部分，你可以通過以下步驟來安裝和運行 Apache JMeter 的做到這一點。

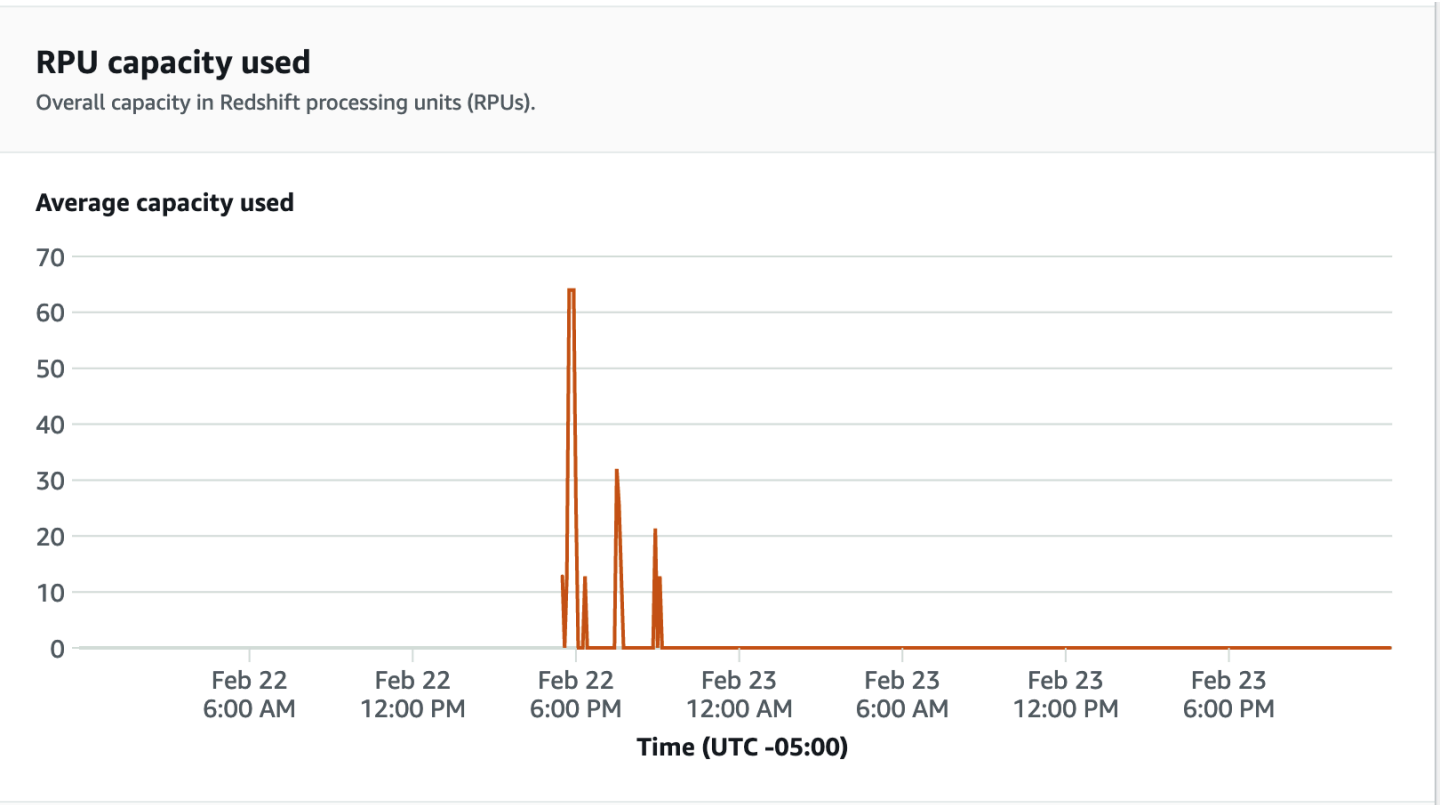
運行使用阿帕奇 JMeter 的負載測試

若要執行負載測試以模擬「N」使用者同時向 Amazon Redshift 提交查詢，您可以使用 Apache JMeter，這是一種以 Java 為基礎的開放原始碼工具。

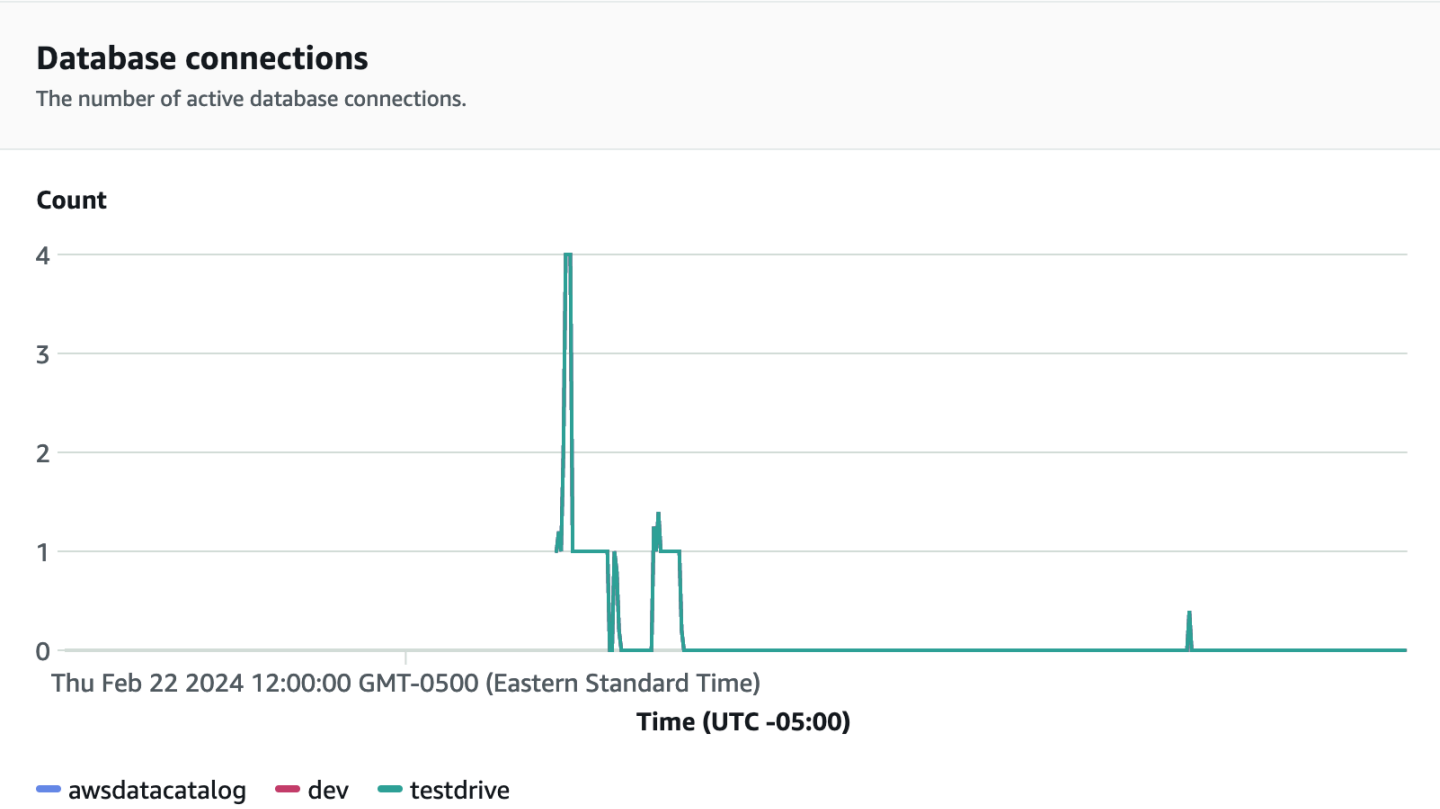
若要安裝和設定 Apache JMeter 以針對 Redshift 無伺服器工作群組執行，請遵循使用分析自動化工具組自動化 Amazon Redshift 負載測試中的說明進行操作。 AWS 它使用AWS 分析自動化工具組 (AAA)，這是一個用於動態部署 Redshift 解決方案的開放原始碼公用程式，自動啟動這些資源。如果您已將自己的資料載入 Amazon Redshift，請務必執行步驟 #5 — 自訂 SQL 選項，以確保您提供想要針對表格測試的適當 SQL 陳述式。使用查詢編輯器 v2 一次測試這些 SQL 陳述式，以確保它們執行時沒有錯誤。

完成自訂 SQL 陳述式並完成測試計劃之後，請針對 Redshift 無伺服器工作群組儲存並執行測試計劃。若要監視測試進度，請開啟 Redshift 無伺服器主控台，瀏覽至 [查詢和資料庫監視]，選擇 [查詢歷程記錄] 索引標籤，然後檢視查詢的相關資訊。

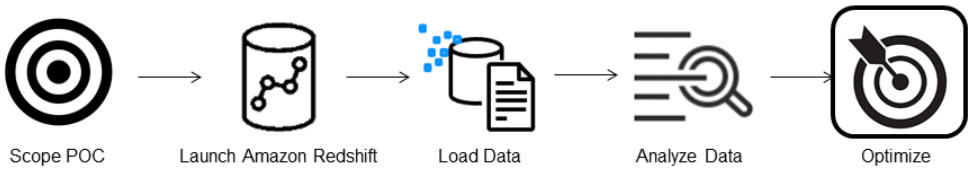
對於效能測量結果，請選擇 Redshift 無伺服器主控台上的「資料庫效能」索引標籤，以監視「資料庫連線」和「CPU 使用率」等指標。您可以在此檢視圖表以監控使用的 RPU 容量，並觀察 Redshift Serverless 如何在工作群組上執行負載測試時自動擴展以滿足並行工作負載需求。



資料庫連線是另一個有用的測量結果，可在執行負載測試時監視，以瞭解工作群組在指定時間如何處理多個並行連線，以滿足日益增加的工作負載需求。



步驟 5：最佳化



Amazon Redshift 透過提供各種組態和功能來支援個別使用案例，讓成千上萬的使用者每天處理 EB 的資料，並強化其分析工作負載。在這些選項之間進行選擇時，客戶正在尋找可協助他們判斷最佳資料倉儲組態以支援 Amazon Redshift 工作負載的工具。

試駕

您可以使用「試用方案」，根據潛在的組態自動重新執行現有工作負載，並分析對應的輸出，以評估要將工作負載移轉至的最佳目標。如需使用試用方案評估不同 Amazon Redshift 組態的相關資訊，請參閱使用 Redshift 試用方案尋找工作負載的最佳 Amazon Redshift 組態。

設計資料表的 Amazon Redshift 最佳實務

在您規劃資料庫時，有些關鍵的資料表設計決策會嚴重影響整體查詢效能。這些設計選擇也會嚴重影響儲存需求，進而因減少 I/O 操作數及處理查詢所需的記憶體而影響查詢效能。

在本節中，您可以找到最重要的設計決策摘要及最佳化查詢效能的最佳實務。使用自動資料表最佳化提供資料表設計選項的詳細說明與範例。

主題

選擇最佳的排序索引鍵

Amazon Redshift 依據排序索引鍵，將您的資料以排序順序儲存於磁碟。Amazon Redshift 查詢最佳化工具使用排序順序來決定最佳查詢計畫。

> **Note**
>
> 當您使用自動資料表最佳化時，您不需要選擇資料表的排序索引鍵。如需詳細資訊，請參閱 使用自動資料表最佳化。

部分最佳做法建議如下：

- 若要讓 Amazon Redshift 選擇適當的排序順序，請為排序索引鍵指定 AUTO。
- 如果會經常查詢較新的資料，請指定時間戳記欄位做為排序索引鍵的前導欄。

  查詢將更有效率，因為它們會跳過位於時間範圍之外的整個區塊。
- 如果您經常對一個欄位進行範圍篩選或相等篩選，請指定該欄位做為排序索引鍵。

  Amazon Redshift 可跳過讀取該欄位的整個資料區塊。它可以這麼做是因為它會追蹤儲存於每個區塊的最小與最大欄位值，並跳過不適用於述詞範圍的區塊。

- 如果您經常聯結資料表，請指定聯結欄位做為排序索引鍵與分佈索引鍵。

  這麼做可以讓查詢最佳化工具選擇排序合併聯結，而非較慢的雜湊聯結。由於資料已經以聯結索引鍵進行排序，查詢最佳化工具可跳過排序合併聯結的排序階段。

選擇最佳的分佈方式

當您執行查詢時，查詢最佳化工具會視需要將資料列重新配送至運算節點，以執行任何聯結與彙總。選擇資料表配送樣式的目的是，藉由在執行查詢之前，將資料配置至需要的位置，以降低重新配送步驟所帶來的影響。

> **Note**
>
> 當您使用自動資料表最佳化時，您不需要選擇資料表的分佈樣式。如需詳細資訊，請參閱 使用自動資料表最佳化。

部分最佳做法建議如下：

1. 將事實資料表與一維度資料表配送至它們的共通欄位。

   您的事實資料表只能有一個分佈索引鍵。聯結其他索引鍵的任何資料表都不會與事實資料表共置。依據其聯結的頻率與聯結資料列的大小，選擇一個維度進行共置。將維度資料表的主索引鍵與事實資料表的對應外部索引鍵指定為 DISTKEY。

2. 依據篩選過的資料集大小，選擇最大的維度。

   只有用於聯結的資料列必須配送，因此應考量篩選之後的資料集大小，而非資料表的大小。

3. 在經過篩選的結果集中選擇高基數的欄位。

   例如，如果您將銷售資料表配送至日期欄位，您可能會獲得相當平均的資料配送，除非大部分的銷售是季節性的。但是，如果您通常使用限制範圍的述詞來篩選以縮小日期期間，則大多數篩選過的資料列會發生在一組有限的分割上，而且查詢工作負載會發生偏斜。

4. 變更一些維度資料表以使用 ALL 分佈。

   如果維度資料表無法與事實資料表或其他重要的聯結資料表共置，您可以藉由將整個資料表發佈至所有節點，以大幅提升查詢效能。使用 ALL 分佈會增加儲存空間要求，並增加載入時間與維護操作，因此在選擇 ALL 分佈之前，您應權衡所有因素。

若要讓 Amazon Redshift 選擇適當的分佈樣式，請指定分佈樣式的 AUTO。

如需選擇分佈樣式的相關資訊，請參閱 使用資料分佈樣式。

讓 COPY 選擇壓縮編碼

當您建立資料表時，可指定壓縮編碼，但在大多數情況下，自動壓縮可產生最佳結果。

ENCODE AUTO 是資料表的預設值。當資料表設定為 ENCODE AUTO 時，Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。如需詳細資訊，請參閱 CREATE TABLE 及 ALTER TABLE。

COPY 命令會在進行載入操作時，自動分析您的資料並將壓縮編碼套用至空資料表。

選擇壓縮編碼時，自動壓縮會平衡整體效能。如果排序索引鍵欄位的壓縮比相同查詢中的其他欄位高出許多，限制範圍的掃描執行效果可能較差。因此，自動壓縮會選擇效率較低的壓縮編碼，以維持排序索引鍵欄位與其他欄位之間的平衡。

假設您資料表的排序索引鍵為日期或時間戳記，而且該資料表使用許多大型 varchar 欄位。在此情況下，如果不壓縮排序索引鍵欄位，您可能會獲得較高的效能。在資料表上執行 ANALYZE COMPRESSION 命令，然後使用編碼以建立新資料表，但省略排序索引鍵的壓縮編碼。

自動壓縮編碼有效能上的成本，但僅限於資料表是空的，而且尚未有壓縮編碼的情況。對於短期資料表及經常建立的資料表，例如臨時資料表，請使用自動壓縮載入資料表一次，或執行 ANALYZE COMPRESSION 命令。然後，使用這些編碼建立新資料表。您可以在 CREATE TABLE 陳述式中新增編碼，或使用 CREATE TABLE LIKE 以相同編碼建立新資料表。

如需詳細資訊，請參閱 利用自動壓縮載入資料表。

定義主索引鍵與外部索引鍵限制條件

在適當的位置定義資料表之間的主索引鍵與外部索引鍵限制條件。儘管它們只是資訊性的，但查詢最佳化工具會使用這些限制條件來產生更有效的查詢計畫。

除非應用程式強制執行限制，否則請勿定義主索引鍵和外部索引鍵限制條件。Amazon Redshift 不會強制執行唯一的主索引鍵和外部索引鍵限制條件。

如需有關 Amazon Redshift 如何使用限制條件的其他資訊，請參閱 定義資料表限制。

使用可能的最小欄位大小

為方便起見，請勿使用最大欄位大小。

反之，舉例而言，請考量您可能在資料欄位中儲存的最大值，並相應地調整其大小。例如，用於存儲郵局使用的美國州和地區縮寫的 CHAR 列只需要是 CHAR (2) 。

在日期欄位中使用日期/時間資料類型

Amazon Redshift 儲存 DATE 與 TIMESTAMP 資料的效率比儲存 CHAR 或 VARCHAR 資料的效率高，這將帶來更好的查詢效能。儲存日期/時間資訊時，依據您所需的解析度，使用 DATE 或 TIMESTAMP 資料類型，而非使用字元類型。如需詳細資訊，請參閱 日期時間 (Datetime) 類型。

載入資料的 Amazon Redshift 最佳實務

主題

載入極大的資料集將非常耗時，並消耗大量運算資源。資料的載入方式也會影響查詢效能。本節介紹使用 COPY 命令、大量插入及臨時資料表，以有效率地載入資料的最佳實務。

參加載入資料教學

教學課程：從 Amazon S3 載入資料 提供完整的逐步教學，介紹如何將資料上傳至 Amazon S3 儲存貯體，然後使用 COPY 命令將資料載入至您的資料表。此教學課程包括協助排解載入錯誤的問題，並比較從單一檔案與從多個檔案載入之間的效能差異。

使用 COPY 命令載入資料

COPY 命令會從 Amazon S3、Amazon EMR、Amazon DynamoDB 或遠端主機的多個資料來源平行載入資料。COPY 載入大量資料的效率高於使用 INSERT 陳述式，儲存資料的效率也比較高。

如需使用 COPY 命令的相關資訊，請參閱 從 Amazon S3 載入資料 與 從 Amazon DynamoDB 資料表載入資料。

使用單一 COPY 命令從多個檔案載入

Amazon Redshift 可以從多個已壓縮的資料檔案平行載入資料。您可以透過使用 Amazon S3 物件字首或使用資訊清單檔案來指定要載入的檔案。

但是，如果您同時使用多個 COPY 命令從多個檔案載入一個資料表，Amazon Redshift 將被迫執行序列化載入。此類型的載入速度非常慢，而且如果資料表有已定義的排序欄位，最後將需要 VACUUM 程序。如需使用 COPY 平行載入資料的相關資訊，請參閱 從 Amazon S3 載入資料。

載入資料檔案

來源資料檔案有不同的格式，並會使用不同的壓縮演算法。使用 COPY 命令載入資料時，Amazon Redshift 會載入 Amazon S3 儲存貯體字首參照的所有檔案。(字首是物件索引鍵名稱開頭的字元字串。) 如果字首參照多個檔案或可以分割的檔案，Amazon Redshift 會利用 Amazon Redshift 的 MPP 架構平行載入資料。這會在叢集中的節點上劃分工作負載。相反地，當您從無法分割的檔案載入資料時，Amazon Redshift 會強制執行序列化載入，而此速度會慢很多。以下各節說明將不同檔案類型載入 Amazon Redshift 的建議方式 (視其格式和壓縮而定)。

從可分割的檔案載入資料

下列檔案可在載入資料時自動分割檔案：

- 未壓縮的 CSV 檔案
- 使用 BZIP 壓縮的 CSV 檔案
- 單欄式檔案 (Parquet/ORC)

Amazon Redshift 會自動將 128MB 或更大的檔案分割成多個區塊。如果單欄式檔案 (特別是 Parquet 和 ORC) 小於 128MB，則不會分割。Redshift 會利用平行運作的分割部分來載入資料。這可提供快速的載入效能。

從部可分割的檔案載入資料

使用其他壓縮演算法 (例如 GZIP) 壓縮時，JSON 或 CSV 等檔案類型不會自動分割。對於這些檔案，我們建議將資料手動分割為多個大小接近的較小文件 (壓縮後的 1 MB 到 1 GB)。此外，檔案數量應為您叢集中分割的倍數。如需如何將資料分割為多個檔案，以及使用 COPY 載入資料範例的相關資訊，請參閱從 Amazon S3 載入資料。

壓縮您的資料檔案

當您想要壓縮大型載入檔案時，我們建議您使用 gzip、lzop、bzip2 或 ZStandard 來壓縮檔案，並將資料分割成多個較小的檔案。

使用 COPY 命令並指定 GZIP、LZOP、BZIP2 或 ZSTD 選項。此範例從縱線分隔 lzop 檔案載入 TIME 資料表。

```
copy time
from 's3://mybucket/data/timerows.lzo'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
lzop
delimiter '|';
```

在某些情況下，您不必分割未壓縮的資料檔案。如需資料分割及使用 COPY 載入資料範例的相關資訊，請參閱 從 Amazon S3 載入資料。

在載入前後驗證資料檔案

從 Amazon S3 載入資料之前，請先確認 Amazon S3 儲存貯體包含所有正確的檔案，而且只包含這些檔案。如需詳細資訊，請參閱 驗證您的儲存貯體中存在正確的檔案。

在載入操作完成後，查詢 STL_LOAD_COMMITS 系統資料表，確認預期的檔案已經載入。如需詳細資訊，請參閱 驗證資料已正確載入。

使用多資料列插入

如果 COPY 命令不是您的選項，而且您需要 SQL 插入，請在可行時使用多資料列插入。當您的資料一次只有一列或幾列時，資料壓縮是沒有效率的。

多資料列插入可批次執行一系列的插入，藉此提升效能。以下範例使用單一 INSERT 陳述式，將三個資料列插入包含四個欄位的資料表。這仍是小型插入，僅用於說明多資料列插入的語法。

```
insert into category_stage values
```

```
(default, default, default, default),
(20, default, 'Country', default),
(21, 'Concerts', 'Rock', default);
```

如需詳細資訊和範例，請參閱 INSERT。

使用大量插入

使用大量插入操作與 SELECT 子句以執行高效率資料插入。

當您必須從一個資料表將資料或資料的子集移至另一個資料表時，請使用 INSERT 與 CREATE TABLE AS 命令。

例如，以下 INSERT 陳述式選取 CATEGORY 資料表中的所有資料列，然後將它們插入 CATEGORY_STAGE 資料表。

```
insert into category_stage
(select * from category);
```

以下範例建立 CATEGORY_STAGE 做為 CATEGORY 的副本，並將 CATEGORY 中的所有資料列插入 CATEGORY_STAGE。

```
create table category_stage as
select * from category;
```

以排序索引鍵順序載入資料

以排序索引鍵的順序載入資料，以避免必須清空。

如果每個批次第新資料皆遵循您資料表中的現有資料列，您的資料將按照排序順序正確儲存，而且無需執行清空。您無需再每次載入時預先排序資料列，因為 COPY 會在載入每個批次的外來資料時進行排序。

例如，您每天根據當天的活動來載入資料。如果您的排序索引鍵為時間戳記欄位，您的資料將按照排序順序儲存。會發生這樣的順序，是因為當天的資料總是會附加至前一天資料的結尾。如需詳細資訊，請參閱 以排序索引鍵順序載入資料。如需清除操作的相關資訊，請參閱清除資料表。

將資料載入循序區塊

如果您需要新增大量資料，將資料依據排序順序載入循序區塊，即可無需執行清除。

例如，假設您需要載入 2017 年 1 月至 2017 年 12 月的活動資料表。假設每個月都在單一檔案中，請載入一月、二月等的資料列。當您載入完成時，資料表已完全排序，無需執行清空。如需詳細資訊，請參閱 使用時間序列資料表。

載入非常大的資料集時，需要排序的空間可能會超過可用的總空間。藉由將資料載入至較小的區塊，可在每次載入時，使用較少的中間排序空間。此外，如果 COPY 失敗並轉返時，載入較小的區塊可使其較容易重新啟動。

使用時間序列資料表

如果您的資料有固定的保留期間，您可以將資料整理為時間序列資料表的順序。在這樣的順序中，每個資料表都是相同的，但包含不同時間範圍的資料。

只要在相應的資料表上執行 DROP TABLE 命令，即可輕鬆移除舊資料。此方法比執行大規模 DELETE 程序更快速，而且後續無需執行 VACUUM 程序以回收空間。若要隱藏資料儲存於不同資料表的事實，您可以建立一個 UNION ALL 檢視。當您刪除舊資料時，強化您的 UNION ALL 檢視即可移除已刪除的資料表。同樣的，當您將新的期間載入至新資料表時，請將新資料表新增至該檢視。若要傳送訊號至最佳化器以跳過掃描與查詢篩選條件不匹配的資料表，您的檢視定義將會篩選對應至每個資料表的日期範圍。

避免在 UNION ALL 檢視中有過多的資料表。每個額外的資料表都會增加一些查詢的處理時間。資料表不需要使用相同的時間範圍。例如，您可能有不同期間的資料表，例如每日、每月及每年。

如果您使用時間序列資料表，並以時間戳記欄位做為排序索引鍵，將可有效地按照排序索引鍵的順序載入資料。如此便不必清除以重新排序資料。如需詳細資訊，請參閱 以排序索引鍵順序載入資料。

排程維護時段

如果排定的維護發生在查詢執行時，查詢將會終止並轉返，您必須予以重新啟動。排定長期執行的操作，例如大型資料載入或 VACUUM 操作，以避免開維護時段。您可以用較小的增量執行資料載入，並管理 VACUUM 操作的大小，藉此降低風險，並且在需要時讓重新啟動更容易。如需詳細資訊，請參閱 將資料載入循序區塊 及 清空資料表。

設計查詢的 Amazon Redshift 最佳實務

為達到最大的查詢效能，在建立查詢時，請遵循這些建議：

- 依據最佳實務設計資料表，為查詢效能提供穩固的基礎。如需詳細資訊，請參閱 設計資料表的 Amazon Redshift 最佳實務。
- 避免使用 `select *`。僅包含您實際需要的欄位。

- 使用 CASE 條件式運算式 執行複雜的彙總，而非從相同的資料表多次選取。
- 除非絕對必要，否則請勿使用交叉聯結。沒有聯結條件的這些聯結，會導致兩個資料表的笛卡兒乘積。交叉聯結通常會執行為巢狀迴圈聯結，這是可能的聯結類型中最慢的。
- 如果查詢中的一個資料表僅用於述詞條件，而且子查詢返回少量資料列 (小於約 200)，則使用子查詢。下列範例使用子查詢以避免聯結 LISTING 資料表。

```
select sum(sales.qtysold)
from sales
where salesid in (select listid from listing where listtime > '2008-12-26');
```

- 使用述詞以盡可能限制資料集。
- 在述詞中，使用最便宜的運算子。比較條件 運算子比 LIKE 運算子好。LIKE 運算子仍比 SIMILAR TO 或 POSIX 運算子 好。
- 避免在查詢述詞中使用函式。使用函式需要大量資料列以解析查詢的中間步驟，因此會提高查詢成本。
- 如有可能，使用 WHERE 子句以限制資料集。接著，查詢規劃器可使用資料列順序，協助判斷哪些記錄符合條件，以跳過掃描大量的磁碟區塊。如果不使用它，查詢執行引擎必須完整掃描參與的欄位。
- 新增述詞以篩選參與聯結的資料表，即使這些述詞套用相同的篩選條件。查詢會傳回相同的結果集，但 Amazon Redshift 可在掃描步驟之前篩選聯結資料表，並有效地跳過掃描這些資料表的區塊。如果您在用於聯結條件的欄位上進行篩選，就不需要備援篩選條件。

  例如，假設您要聯結 SALES 與 LISTING 以搜尋 12 月之後列出之門票的門票銷售，依照銷售者分組。這兩個資料表皆以日期排序。以下查詢聯結資料表上的共同索引鍵，並篩選大於 12 月 1 日的 `listing.listtime` 值。

```
select listing.sellerid, sum(sales.qtysold)
from sales, listing
where sales.salesid = listing.listid
and listing.listtime > '2008-12-01'
group by 1 order by 1;
```

  WHERE 子句不包含 `sales.saletime` 的述詞，因此執行引擎被迫掃描整個 SALES 資料表。如果您知道篩選條件會導致較少的資料列參與聯結，那麼也請加入該篩選條件。下列範例大幅縮短執行時間。

```
select listing.sellerid, sum(sales.qtysold)
```

```
from sales, listing
where sales.salesid = listing.listid
and listing.listtime > '2008-12-01'
and sales.saletime > '2008-12-01'
group by 1 order by 1;
```

- 在 GROUP BY 子句中使用排序索引鍵，讓查詢規劃器可以使用更有效率的彙總。當查詢的 GROUP BY 列表僅包含排序索引鍵欄位，而且其中一欄也是分佈索引鍵，則該查詢可能有資格進行單一階段彙總。GROUP BY 列表中的排序索引鍵欄位必須包含第一個排序索引鍵，然後是您要依照排序索引鍵順序使用的其他排序索引鍵。例如，使用第一個排序索引鍵、第一和第二個排序索引鍵，第一、二和第三個排序索引鍵等等，都是有效的。使用第一和第三個排序索引鍵是無效的。

  您可以執行 EXPLAIN 命令並在查詢的彙總步驟尋找 `XN GroupAggregate`，以確認使用單一階段彙總。

- 如果您同時使用 GROUP BY 與 ORDER BY 子句，請確定兩個欄位的順序是相同的。亦即，使用以下方法。

  ```
  group by a, b, c
  order by a, b, c
  ```

  請勿使用以下方法。

  ```
  group by b, c, a
  order by a, b, c
  ```

使用 Amazon Redshift Advisor 的建議

為協助您提高 Amazon Redshift 叢集的效能並降低營運成本，Amazon Redshift Advisor 提供您有關進行變更的具體建議。Advisor 透過分析叢集的效能與用量指標來開發其自訂的建議。這些量身訂製的建議與操作及叢集設定有關。為協助您排定最佳化的優先順序，Advisor 會依據影響的高低順序將建議項目進行排名。

Advisor 的建議以觀察相關效能統計與操作資料為基礎。Advisor 藉由在叢集上執行測試來判斷是否有測試值在指定範圍內，進而開發出觀察結果。如果測試結果在範圍之外，Advisor 將為您的叢集產生觀察結果。同時，Advisor 會建立建議，告訴您如何將觀察到的值恢復至最佳實務的範圍。Advisor 只會顯示將為效能與操作帶來大幅影響的建議。當 Advisor 判斷建議已獲得處理，便會從建議清單中移除該建議。

例如，假設您的資料倉儲包含大量未壓縮的資料表欄位。在此情況下，您可以使用 ENCODE 參數指定欄壓縮以重建資料表，以節省叢集儲存空間成本。在另一個範例中，假設 Advisor 觀察到您的叢集在未壓縮的資料表資料中包含大量資料。在此情況下，它將提供 SQL 程式碼區塊，以尋找可做為壓縮候選的資料表欄，並提供說明如何壓縮這些欄的資源。

Amazon Redshift 區域

Amazon Redshift 建議程式功能僅適用於下列 AWS 區域：

- 美國東部 (維吉尼亞北部) 區域 (us-east-1)
- 美國東部 (俄亥俄) 區域 (us-east-2)
- 美國西部 (加利佛尼亞北部) 區域 (us-west-1)
- 美國西部 (奧勒岡) 區域 (us-west-2)
- 非洲 (開普敦) 區域 (af-south-1)
- 亞太區域 (香港) 區域 (ap-east-1)
- 亞太區域 (海德拉巴) 區域 (ap-south-2)
- 亞太區域 (雅加達) 區域 (ap-southeast-3)
- 亞太區域 (墨爾本) 區域 (ap-southeast-4)
- 亞太區域 (孟買) 區域 (ap-south-1)
- 亞太區域 (大阪) (ap-northeast-3)
- 亞太區域 (首爾) 區域 (ap-northeast-2)
- 亞太區域 (新加坡) 區域 (ap-southeast-1)
- 亞太區域 (雪梨) 區域 (ap-southeast-2)
- 亞太區域 (東京) 區域 (ap-northeast-1)
- 加拿大 (中部) 區域 (ca-central-1)
- 加拿大西部 (卡加利) 區域 (ca-west-1)
- 中國 (北京) 區域 (cn-north-1)
- 中國 (寧夏) 區域 (cn-northwest-1)
- 歐洲 (法蘭克福) 區域 (eu-central-1)
- 歐洲 (愛爾蘭) 區域 (eu-west-1)
- 歐洲 (倫敦) 區域 (eu-west-2)
- 歐洲 (米蘭) 區域 (eu-south-1)

- 歐洲 (巴黎) 區域 (eu-west-3)
- 歐洲 (西班牙) 區域 (eu-south-2)
- 歐洲 (斯德哥爾摩) 區域 (eu-north-1)
- 歐洲 (蘇黎世) 區域 (eu-central-2)
- 以色列 (特拉維夫) 區域 (il-central-1)
- 中東 (巴林) 區域 (me-south-1)
- 中東 (阿拉伯聯合大公國) 區域 (me-central-1)
- 南美洲 (聖保羅) 區域 (sa-east-1)

主題

檢視 Amazon Redshift 顧問建議

您可以使用 Amazon Redshift 馬 Amazon Redshift 主控台、Amazon Redshift API 或. AWS CLI若要存取建議，您必須擁有 IAM 角色或身分`redshift:ListRecommendations`附加的權限。

在 Amazon Redshift 佈建的主控台上檢視 Amazon Redshift 顧問建議

您可以在上檢視 Amazon Redshift 顧問建議。 AWS Management Console

在主控台上檢視針對 Amazon Redshift 馬遜紅移叢集的亞馬遜紅移顧問建議

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單中，選擇 Advisor。
3. 展開建議事項以檢視更多詳細資訊。在此頁面上，您可以排序和分組建議。

使用 Amazon Redshift API 操作查看 Amazon Redshift 顧問建議

您可以使用 Amazon Redshift API 列出適用於 Amazon Redshift 叢集的 Amazon Redshift 顧問建議。通常，您可以使用自己選擇的程式設計語言來開發和應用，以便使用 AWS SDK 呼叫 `redshift:ListRecommendations` API。如需詳細資訊，請參閱ListRecommendations亞 Amazon Redshift API 參考中的一文。

使用操作檢視 Amazon Redshift 顧問建 AWS Command Line Interface 議

您可以使用列出 Amazon Redshift 集群的 Amazon Redshift 顧問建議。 AWS Command Line Interface若要取得更多資訊，請參閱《AWS CLI 指令參考》中的清單建議。

Amazon Redshift Advisor 建議

Amazon Redshift Advisor 提供有關如何最佳化 Amazon Redshift 叢集以提高效能並節省營運成本的建議。如前所述，您可以在主控台找到各項建議的解釋。您可以在以下章節中找到這些建議的進一步詳細資訊。

主題

- 壓縮以 COPY 載入的 Amazon S3 檔案物件
- 隔離多個作用中資料庫
- 重新配置工作負載管理 (WLM) 記憶體
- 在 COPY 期間跳過壓縮分析
- 分割由 COPY 載入的 Amazon S3 物件
- 更新資料表統計資訊
- 啟用短期查詢加速
- 更改資料表上的分佈索引鍵
- 變更資料表上的排序索引鍵
- 修改資料欄上的壓縮編碼
- 資料類型建議

壓縮以 COPY 載入的 Amazon S3 檔案物件

COPY 命令利用 Amazon Redshift 中的大規模平行處理 (MPP) 架構，以同時讀取與載入資料。它可從 Amazon S3、DynamoDB 資料表讀取檔案，以及從一或多個遠端主機讀取文字輸出。

載入大量資料時，強烈建議您使用 COPY 命令從 S3 載入已壓縮的資料檔案。壓縮大型資料集可節省上傳檔案至 Amazon S3 的時間。COPY 亦可在讀取檔案時解壓縮檔案，以加速載入程序。

分析

載入大型未壓縮資料集的長時間執行 COPY 命令，通常有機會大幅提升效能。Advisor 分析可識別載入大型未壓縮資料集的 COPY 命令。在此情況下，Advisor 會產生在 Amazon S3 中的來源檔案實作壓縮的建議。

建議

確定載入大量資料或長時間執行的每個 COPY 皆從 Amazon S3 擷取已壓縮的資料物件。您可以用超級使用者的身分執行以下 SQL 命令，識別從 Amazon S3 載入大型未壓縮資料集的 COPY。

```
SELECT
    wq.userid, query, exec_start_time AS starttime, COUNT(*) num_files,
    ROUND(MAX(wq.total_exec_time/1000000.0),2) execution_secs,
    ROUND(SUM(transfer_size)/(1024.0*1024.0),2) total_mb,
    SUBSTRING(querytxt,1,60) copy_sql
FROM stl_s3client s
JOIN stl_query q USING (query)
JOIN stl_wlm_query wq USING (query)
WHERE s.userid>1 AND http_method = 'GET'
    AND POSITION('COPY ANALYZE' IN querytxt) = 0
    AND aborted = 0 AND final_state='Completed'
GROUP BY 1, 2, 3, 7
HAVING SUM(transfer_size) = SUM(data_size)
AND SUM(transfer_size)/(1024*1024) >= 5
ORDER BY 6 DESC, 5 DESC;
```

如果在您載入資料之後，暫存資料仍留存在 Amazon S3 (資料湖經常出現此情況)，以壓縮形式儲存此資料可降低您的儲存空間成本。

實作提示

- 壓縮之後的理想物件大小介於 1–128 MB 之間。
- 您可以用 gzip、lzop 或 bzip2 格式壓縮檔案。

隔離多個作用中資料庫

根據最佳實務，建議您隔離 Amazon Redshift 中的各個資料庫。查詢執行於特定資料庫，而且無法存取叢集上其他資料庫的資料。但是，您在叢集上所有資料庫執行的查詢，將共用相同的底層叢集儲存空間與運算資源。當單一叢集包含多個使用中資料庫時，其工作負載通常是不相關的。

分析

Advisor 分析會檢閱叢集上的所有資料庫，以找出同時執行的使用中工作負載。如果有同時執行的使用中工作負載，Advisor 分析會產生建議，建議您考慮將資料庫遷移至個別的 Amazon Redshift 叢集。

建議

考慮將經常被查詢的資料庫移至個別的專屬叢集。使用個別的叢集可減少資源爭奪並提升查詢效能。這是因為它能讓您設定各個叢集的儲存空間大小、成本，以及各個工作負載所需的效能。同時，不相關的工作負載通常會受益於不同的工作負載管理組態。

若要找出最常用的資料庫，請以超級使用者身分執行此 SQL 命令。

```
SELECT database,
  COUNT(*) as num_queries,
  AVG(DATEDIFF(sec,starttime,endtime)) avg_duration,
  MIN(starttime) as oldest_ts,
  MAX(endtime) as latest_ts
FROM stl_query
WHERE userid > 1
GROUP BY database;
```

實作提示

- 因為使用者必須特別連接至各個資料庫，而查詢只能存取單一資料庫，因此將資料庫移至個別叢集對使用者的影響很小。
- 移動資料庫的選項之一是執行以下步驟：
  1. 暫時將目前叢集的快照還原至相同大小的叢集。
  2. 從新的叢集刪除所有資料庫，要移動的目標資料庫除外。
  3. 調整叢集大小以符合適當的節點類型，並計算資料庫的工作負載。

重新配置工作負載管理 (WLM) 記憶體

Amazon Redshift 將使用者查詢路由到 實作手動 WLM 以進行處理。工作負載管理 (WLM) 定義如何將這些查詢路由到佇列。Amazon Redshift 會為每個佇列都配置叢集的一部分可用記憶體。佇列的記憶體再分配給佇列的查詢槽。

當佇列設定的插槽多於工作負載的需要時，配置到這些未使用插槽的記憶體將無法充分利用。減少設定的插槽以符合尖峰工作負載需求，可將未使用的記憶體重新分配給使用中的插槽，並藉此提升查詢效能。

分析

Advisor 分析將檢閱工作負載並行需求，以識別具有未使用插槽的查詢佇列。Advisor 發現以下情形時將會產生建議，建議您降低佇列中的插槽數量：

- 在整個分析過程中有完全未使用之插槽的佇列。
- 擁有四個以上插槽，而且在整個分析過程中至少有兩個未使用插槽的佇列。

建議

減少設定的插槽以符合尖峰工作負載需求，可將未使用的記憶體重新分配給使用中的插槽。針對其插槽不曾完全充分利用的佇列，考慮減少其設定的插槽數量。若要找出這些佇列，您可以用超級使用者的身分執行以下 SQL 命令，比較各個佇列的每小時尖峰插槽需求。

```
WITH
 generate_dt_series AS (select sysdate - (n * interval '5 second') as dt from (select
 row_number() over () as n from stl_scan limit 17280)),
 apex AS (
     SELECT iq.dt, iq.service_class, iq.num_query_tasks, count(iq.slot_count) as
 service_class_queries, sum(iq.slot_count) as service_class_slots
     FROM
         (select gds.dt, wq.service_class, wscc.num_query_tasks, wq.slot_count
         FROM stl_wlm_query wq
         JOIN stv_wlm_service_class_config wscc ON (wscc.service_class =
 wq.service_class AND wscc.service_class > 5)
         JOIN generate_dt_series gds ON (wq.service_class_start_time <= gds.dt AND
 wq.service_class_end_time > gds.dt)
         WHERE wq.userid > 1 AND wq.service_class > 5) iq
     GROUP BY iq.dt, iq.service_class, iq.num_query_tasks),
     maxes as (SELECT apex.service_class, trunc(apex.dt) as d, date_part(h,apex.dt) as
 dt_h, max(service_class_slots) max_service_class_slots
                     from apex group by apex.service_class, apex.dt,
 date_part(h,apex.dt))
SELECT apex.service_class - 5 AS queue, apex.service_class, apex.num_query_tasks AS
 max_wlm_concurrency, maxes.d AS day, maxes.dt_h || ':00 - ' || maxes.dt_h || ':59' as
 hour, MAX(apex.service_class_slots) as max_service_class_slots
FROM apex
JOIN maxes ON (apex.service_class = maxes.service_class AND apex.service_class_slots =
 maxes.max_service_class_slots)
GROUP BY  apex.service_class, apex.num_query_tasks, maxes.d, maxes.dt_h
ORDER BY apex.service_class, maxes.d, maxes.dt_h;
```

max_service_class_slots 欄位表示該小時查詢佇列中 WLM 查詢插槽的最大數量。如有未利用的佇列，可修改參數群組以實作插槽減量最佳化，如 Amazon Redshift 管理指南所述。

實作提示

- 如果您的工作負載量變化很大，請確定分析有擷取到尖峰使用期間。如果變化不大，請重複執行先前的 SQL 以監控尖峰並行需求。
- 如需有關解譯上述 SQL 程式碼之查詢結果的詳細資訊，請參閱中的 wlm_apex_hourly.sql 指令碼。GitHub

在 COPY 期間跳過壓縮分析

當您以 COPY 命令宣告的壓縮編碼，將資料載入空的資料表時，Amazon Redshift 將會套用儲存空間壓縮。此最佳化可確保叢集中的資料有效地儲存，即使是由最終使用者載入。需要套用壓縮的分析可能需要很長的時間。

分析

Advisor 分析會檢查因自動壓縮分析而延遲的 COPY 操作。此分析會在資料載入時取樣資料以判斷壓縮編碼。此取樣類似 ANALYZE COMPRESSION 命令所執行的取樣。

當您在結構化程序中載入資料時，例如隔夜擷取、轉換、載入 (ETL) 批次，您可以預先定義壓縮。您也可以最佳化您的資料表定義，以永久跳過此階段，而不產生任何負面影響。

建議

為藉由跳過壓縮分析階段以提升 COPY 反應速度，請實作以下兩個選項之一：

- 當您建立使用 COPY 命令載入的資料表時，請使用欄位 ENCODE 參數。
- 在 COPY 命令中套用 COMPUPDATE OFF 參數，以一併關閉壓縮。

最佳解決方案通常是在資料表建立期間使用欄位編碼，因為此方法也同時保有將壓縮資料儲存於磁碟的好處。您可以使用 ANALYZE COMPRESSION 命令以建議壓縮編碼，但您必須重建資料表以套用這些編碼。若要自動執行此程序，您可以使 AWSColumnEncoding用上的公用程式 GitHub。

若要識別最近觸發自動壓縮分析的 COPY 操作，請執行以下 SQL 命令。

```
WITH xids AS (
  SELECT xid FROM stl_query WHERE userid>1 AND aborted=0
```

```
    AND querytxt = 'analyze compression phase 1' GROUP BY xid
    INTERSECT SELECT xid FROM stl_commit_stats WHERE node=-1)
SELECT a.userid, a.query, a.xid, a.starttime, b.complyze_sec,
    a.copy_sec, a.copy_sql
FROM (SELECT q.userid, q.query, q.xid, date_trunc('s',q.starttime)
    starttime, substring(querytxt,1,100) as copy_sql,
    ROUND(datediff(ms,starttime,endtime)::numeric / 1000.0, 2) copy_sec
    FROM stl_query q JOIN xids USING (xid)
    WHERE (querytxt ilike 'copy %from%' OR querytxt ilike '% copy %from%')
    AND querytxt not like 'COPY ANALYZE %') a
LEFT JOIN (SELECT xid,
    ROUND(sum(datediff(ms,starttime,endtime))::numeric / 1000.0,2) complyze_sec
    FROM stl_query q JOIN xids USING (xid)
    WHERE (querytxt like 'COPY ANALYZE %'
    OR querytxt like 'analyze compression phase %')
    GROUP BY xid ) b ON a.xid = b.xid
WHERE b.complyze_sec IS NOT NULL ORDER BY a.copy_sql, a.starttime;
```

實作提示

- 請確定在您 ETL 程序中建立的所有大容量資料表 (例如，臨時資料表與暫存資料表) 宣告所有欄位的壓縮編碼 (第一個排序索引鍵除外)。
- 估算前述 SQL 命令識別之各個 COPY 命令所載入資料表的預期生命週期大小。如果您有信心資料表將維持極小，請以 COMPUPDATE OFF 參數一併關閉壓縮。否則，請在以 COPY 命令載入資料之前，以明確的壓縮建立資料表。

分割由 COPY 載入的 Amazon S3 物件

COPY 命令利用 Amazon Redshift 中的大規模平行處理 (MPP) 架構，從 Amazon S3 的檔案讀取與載入資料。COPY 命令從多個檔案平行載入資料，並將工作負載分散至您叢集中的各個節點。若要達到最佳傳輸量，強烈建議您將資料分割為多個檔案來善加利用平行處理。

分析

Advisor 分析可識別載入包含於暫存在 Amazon S3 少量檔案中的大型資料集的 COPY 命令。從少量檔案載入大型資料集的長時間執行 COPY 命令，通常有機會大幅提升效能。當 Advisor 發現這些 COPY 命令花費大量時間，將會建立建議，藉由將資料分割至 Amazon S3 中的其他檔案以增加平行處理。

建議

在此情況下，建議採取以下依照優先順序排序的動作：

1. 最佳化載入檔案數量少於叢集節點數量的 COPY 命令。
2. 最佳化載入檔案數量少於叢集分割數量的 COPY 命令。
3. 最佳化檔案數量不是叢集分割數量倍數的 COPY 命令。

某些 COPY 命令會載入大量資料或長時間執行。對於這些命令，建議您從 Amazon S3 載入資料物件的數量等於叢集中分割數量的倍數。若要識別每個 COPY 命令已載入多少個 S3 物件，請以超級使用者身分執行以下 SQL 程式碼。

```
SELECT
    query, COUNT(*) num_files,
    ROUND(MAX(wq.total_exec_time/1000000.0),2) execution_secs,
    ROUND(SUM(transfer_size)/(1024.0*1024.0),2) total_mb,
    SUBSTRING(querytxt,1,60) copy_sql
FROM stl_s3client s
JOIN stl_query q USING (query)
JOIN stl_wlm_query wq USING (query)
WHERE s.userid>1 AND http_method = 'GET'
    AND POSITION('COPY ANALYZE' IN querytxt) = 0
    AND aborted = 0 AND final_state='Completed'
GROUP BY query, querytxt
HAVING (SUM(transfer_size)/(1024*1024))/COUNT(*) >= 2
ORDER BY CASE
WHEN COUNT(*) < (SELECT max(node)+1 FROM stv_slices) THEN 1
WHEN COUNT(*) < (SELECT COUNT(*) FROM stv_slices WHERE node=0) THEN 2
ELSE 2+((COUNT(*) % (SELECT COUNT(*) FROM stv_slices))/(SELECT COUNT(*)::DECIMAL FROM
 stv_slices))
END, (SUM(transfer_size)/(1024.0*1024.0))/COUNT(*) DESC;
```

實作提示

- 節點中的分割數目取決於叢集的節點大小。如需各個節點類型有多少分割的相關資訊，請參閱《Amazon Redshift 管理指南》中的 < Amazon Redshift 中的叢集和節點 > 。
- 您可以透過為集合指定通用字首或字首索引鍵，或在資訊清單檔案中明確列出檔案來載入多個檔案。如需載入檔案的相關資訊，請參閱 從已壓縮和未壓縮的檔案載入資料。

- Amazon Redshift 在分割工作負載時，不會考量檔案大小。分割您的載入資料，使檔案皆有相同的大小，在壓縮之後介於 1 MB 至 1 GB。

更新資料表統計資訊

Amazon Redshift 使用成本型查詢最佳化工具來為查詢選擇最佳執行計畫。成本估算是根據使用 ANALYZE 命令產生的資料表統計資訊。當統計資訊已過時或遺失時，資料庫可能會選擇效率較低的查詢執行計畫，特別是複雜的查詢。維護最新統計資訊有助於複雜的查詢以可能的最短時間執行。

分析

「建議程式」分析會追蹤統計資料為 out-of-date 或遺失的表格。它會檢閱與複雜查詢相關聯的資料表存取中繼資料。如果經常以複雜模式存取的資料表遺失統計資訊，Advisor 會建立重大建議以執行 ANALYZE。如果經常使用複雜模式存取的表格具有 out-of-date 統計資料，「建議程式」會建立建議來執行 ANALYZE。

建議

無論何時，當資料表內容有大幅變更時，請以 ANALYZE 更新統計資訊。當有大量新資料列以 COPY 或 INSERT 命令載入至現有資料表時，我們建議執行 ANALYZE。當有大量資料列以 UPDATE 或 DELETE 命令進行修改時，我們也建議執行 ANALYZE。若要識別遺漏的資料表或 out-of-date 統計資料，請以超級使用者身分執行下列 SQL 命令。其結果將從最大到最小的資料表進行排序。

若要識別遺漏的資料表或 out-of-date 統計資料，請以超級使用者身分執行下列 SQL 命令。其結果將從最大到最小的資料表進行排序。

```
SELECT
   ti.schema||'.'||ti."table" tablename,
   ti.size table_size_mb,
   ti.stats_off statistics_accuracy
 FROM svv_table_info ti
 WHERE ti.stats_off > 5.00
 ORDER BY ti.size DESC;
```

實作提示

預設 ANALYZE 閾值為 10%。此預設值表示特定資料表自上一次 ANALYZE 後，如果變更的列數少於 10%，則 ANALYZE 命令會略過此資料表。因此，您可以選擇在每個 ETL 程序結尾時發出 ANALYZE 命令。採取此方法意味著通常會跳過 ANALYZE，但也能確保 ANALYZE 會在需要時執行。

ANALYZE 統計資訊對於聯結中使用的欄位 (例如，JOIN tbl_a ON col_b) 或做為述詞的欄位 (例如，WHERE col_b = 'xyz') 最有影響。根據預設，ANALYZE 會收集指定資料表中所有欄位的統計資訊。如有需要，您可以針對 ANALYZE 最有影響的欄位來執行 ANALYZE，以減少需要執行 ANALYZE 的時間。您可執行下列 SQL 命令來識別做為述詞使用的欄位。您也可以指定 ANALYZE PREDICATE COLUMNS，讓 Amazon Redshift 選擇要分析哪些欄。

```
WITH predicate_column_info as (
SELECT ns.nspname AS schema_name, c.relname AS table_name, a.attnum as col_num,
 a.attname as col_name,
        CASE
            WHEN 10002 = s.stakind1 THEN array_to_string(stavalues1, '||')
            WHEN 10002 = s.stakind2 THEN array_to_string(stavalues2, '||')
            WHEN 10002 = s.stakind3 THEN array_to_string(stavalues3, '||')
            WHEN 10002 = s.stakind4 THEN array_to_string(stavalues4, '||')
            ELSE NULL::varchar
        END AS pred_ts
   FROM pg_statistic s
   JOIN pg_class c ON c.oid = s.starelid
   JOIN pg_namespace ns ON c.relnamespace = ns.oid
   JOIN pg_attribute a ON c.oid = a.attrelid AND a.attnum = s.staattnum)
SELECT schema_name, table_name, col_num, col_name,
       pred_ts NOT LIKE '2000-01-01%' AS is_predicate,
       CASE WHEN pred_ts NOT LIKE '2000-01-01%' THEN (split_part(pred_ts,
 '||',1))::timestamp ELSE NULL::timestamp END as first_predicate_use,
       CASE WHEN pred_ts NOT LIKE '%||2000-01-01%' THEN (split_part(pred_ts,
 '||',2))::timestamp ELSE NULL::timestamp END as last_analyze
FROM predicate_column_info;
```

如需詳細資訊，請參閱 分析資料表。

啟用短期查詢加速

短期查詢加速 (SQA) 可排定讓短期執行的查詢優先於長期執行的查詢。SQA 會在專用空間中執行短期查詢，所以 SQA 查詢不會被迫在佇列中排在長期查詢後面等待。SQA 只優先處理短期執行且位於使用者定義佇列中的查詢。SQA 可讓短期執行的查詢更快開始執行，使用者會更快看到結果。

如果開啟 SQA，您可以減少或去除短期查詢執行專用的工作負載管理 (WLM) 佇列。此外，長期執行的查詢不需要與短期查詢爭奪佇列中的槽，因此，您可以將 WLM 佇列設定為使用較少的查詢槽。使用較低的並行性時，就大部分工作負載而言，查詢傳輸量會增加，且整體系統效能會改善。如需詳細資訊，請參閱 使用短期查詢加速。

分析

Advisor 會檢查工作負載模式，並報告 SQA 可減少延遲的最近查詢數目，以及符合 SQA 資格的查詢的每日佇列時間。

建議

修改 WLM 組態以開啟 SQA。Amazon Redshift 採用機器學習演算法來分析每一個合格查詢。隨著 SQA 從查詢模式中學習，預測會越準確。如需詳細資訊，請參閱設定工作負載管理。

開啟 SQA 時，根據預設，WLM 會將短期查詢的最長執行期設為動態。建議維持 SQA 最長執行時間的動態設定。

實作提示

若要檢查是否已開啟 SQA，請執行下列查詢。如果查詢傳回一列，表示 SQA 已開啟。

```
select * from stv_wlm_service_class_config
where service_class = 14;
```

如需詳細資訊，請參閱 監控 SQA。

更改資料表上的分佈索引鍵

Amazon Redshift 會根據資料表分佈樣式將資料表列分佈至整個叢集。含有 KEY 分佈的資料表需要將一個資料欄做為分佈索引鍵 (DISTKEY)，而系統會根據 DISTKEY 資料欄值來指派資料表列給叢集的節點分割。

適當的 DISTKEY 會在每個節點分割上放置差不多數量的資料列，且聯結條件經常會參考該資料欄。當系統在 DISTKEY 資料欄上聯結資料表時，就會產生經過最佳化的聯結，以加速查詢效能。

分析

Advisor 會分析叢集的工作負載以找出最適合資料表的分佈索引鍵，其可從 KEY 分佈樣式獲得諸多優勢。

建議

Advisor 所提供的 ALTER TABLE 陳述式可根據分析結果來更改資料表的 DISTSTYLE 和 DISTKEY。請確保在建議群組內實作所有 SQL 陳述式，以發揮顯著的效能優勢。

使用 ALTER TABLE 重新分佈大型資料表會耗用叢集資源，且會不定時鎖定暫時資料表。因此，請在其他叢集工作負載較輕時實作每個建議群組。如需最佳化表格分佈屬性的詳細資訊，請參閱 Amazon Redshift 工程的進階資料表設計操作手冊：分佈樣式與分佈索引鍵。

如需 ALTER DISTSYLE 和 DISTKEY 的相關資訊，請參閱 ALTER TABLE。

> **Note**
>
> 如果沒有看到建議，並不一定表示目前的發佈樣式是最合適的。如果沒有足夠的資料或重新分配的預期益處很小，Advisor 不會提供建議。
> Advisor 建議適用於特定資料表，且不一定適用於含相同資料欄名稱的資料表。欄位名稱相同的資料表，其各欄特性仍可能不同，除非資料表中的資料也相同。
> 如果您看到由 ETL 任務建立或刪除的暫存資料表建議，則需修改 ETL 程序以使用 Advisor 建議的分佈索引鍵。

變更資料表上的排序索引鍵

Amazon Redshift 會根據資料表的排序索引鍵對表格列進行排序。表格列會根據排序鍵欄值進行排序。

透過適當的排序索引鍵排序表格可以加速查詢效能，特別是有範圍限制述詞的查詢，因為需要從磁碟讀取的表格區塊較少。

分析

Advisor 會分析您叢集在數天的工作負載，為您的表格找出能提供助益的排序索引鍵。

建議

Advisor 會提供兩組 ALTER TABLE 陳述式，此陳述式會根據分析來變更排序索引鍵：

- 修改目前沒有排序索引鍵的資料表以新增 COMPECT 排序索引鍵的陳述式。
- 將排序索引鍵從 INTERLEAVED 變更為 COMPOUND 或無排序索引鍵的陳述式。

  使用複合排序索引鍵可大幅降低維護額外負荷。使用複合排序索引鍵的資料表不需要交錯排序所需的高成本 VACUUM REINDEX 操作。實際上，就大多數 Amazon Redshift 工作負載而言，複合排序索引鍵比交錯排序索引鍵更有效果。但是，如果資料表很小，則不使用排序索引鍵以避免排序索引鍵儲存空間負荷會更有效率。

使用 ALTER TABLE 排序大型表格時會消耗叢集資源，您需要在各種不同時間鎖定表格。當叢集的工作負載不高時，請實作每個建議。如需最佳化表格排序索引鍵組態的詳細資訊，請參閱 Amazon Redshift 工程的進階資料表設計操作手冊：複合和交錯排序索引鍵。

如需 ALTER SORTKEY 的相關資訊，請參閱ALTER TABLE。

> **Note**
>
> 如果您沒有看到表格的建議，這並不一定表示當前配置是最好的。當沒有足夠的資料或排序的預期益處不大時，Advisor 不會提供建議。
> Advisor 建議適用於特定表格，且不一定適用於含有相同名稱和資料類型的表格。根據表格中的資料和工作負載，共用欄名稱的表格可能會有不同的建議。

修改資料欄上的壓縮編碼

壓縮是欄層級操作，可降低資料儲存時的大小。Amazon Redshift 中會使用壓縮來節省儲存空間並透過減少磁碟 I/O 量來改善查詢效能。我們建議您根據資料類型和查詢模式為每個資料欄採用最佳壓縮編碼。透過最佳壓縮，查詢可以更有效率地執行，而且資料庫會佔用最少的儲存空間。

分析

Advisor 會持續執行叢集工作負載和資料庫結構描述的分析，以識別每個資料表資料欄的最佳壓縮編碼。

建議

Advisor 會提供 ALTER TABLE 陳述式，以根據其分析來變更特定資料欄的壓縮編碼。

以 ALTER TABLE 變更資料欄壓縮編碼會耗用叢集資源，且需要在不同時間鎖定資料表。建議您在叢集工作負載很輕時實作建議內容。

作為參考，ALTER TABLE 範例 會顯示數個變更資料欄編碼的陳述式。

> **Note**
>
> 當沒有足夠的資料或變更編碼的預期益處不大時，Advisor 不會提供建議。

資料類型建議

Amazon Redshift 具有適用於各種使用案例的 SQL 資料類型程式庫。這些包括整數類型 (例如 `INT`) 和儲存字元的類型 (例如 `VARCHAR`)。Redshift 會以最佳化的方式儲存類型，以提供快速存取和良好的查詢效能。此外，Redshift 還提供用於特定類型的函數，您可以使用這些函數對查詢結果進行格式化或執行計算。

分析

Advisor 會持續執行叢集工作負載和資料庫結構描述的分析，以識別可從資料類型變更中顯著受益的資料欄。

建議

Advisor 提供的 ALTER TABLE 陳述式會新增包含建議資料類型的新資料欄。隨附的 UPDATE 陳述式會將資料從現有資料欄複製到新資料欄。建立新資料欄並載入資料後，請變更查詢和擷取指令碼以存取新資料欄。然後利用專門用於新資料類型的功能和函數 (可在 SQL 函數參考 中找到)。

將現有資料複製到新資料欄可能需要一些時間。建議您在叢集的工作負載較輕時，實作每個建議程式建議。在 資料類型 中參考可用資料類型的清單。

請注意，當沒有足夠的資料或變更資料類型的預期益處不大時，Advisor 不會提供建議。

Amazon Redshift 的教學課程

請遵循這些教學課程中的步驟以瞭解 Amazon Redshift 功能：

- 教學課程：從 Amazon S3 載入資料
- 教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料
- 教學：設定手動工作負載管理 (WLM) 佇列
- 教學課程：將空間 SQL 函數與 Amazon Redshift 搭配使用
- Amazon Redshift ML 的教學課程

使用自動資料表最佳化

自動資料表最佳化是一項自我調整功能，可透過套用排序和分佈索引鍵來自動最佳化資料表的設計，而無須管理員介入。透過使用自動化來調整資料表的設計，您可以快速開始並獲得最快的效能，而無需花費時間來手動調整和實作資料表最佳化。

自動資料表最佳化會持續觀察查詢與資料表的互動方式。其使用進階人工智慧方法來選擇排序和分佈索引鍵，以最佳化叢集工作負載的效能。如果 Amazon Redshift 判定套用索引鍵可改善叢集效能，則資料表會在叢集建立後的數小時內自動進行修改，而且會盡可能降低對查詢造成的影響。

為了充分利用此自動化功能，Amazon Redshift 管理員會建立新資料表，或修改現有資料表以使其能夠使用自動最佳化。分佈樣式或排序索引鍵為 AUTO 的現有資料表已啟用自動化功能。當您對這些資料表執行查詢時，Amazon Redshift 會判斷排序索引鍵或分佈索引鍵是否能改善效能。如果是的話，Amazon Redshift 就會自動修改資料表，而不需要管理員介入。如果執行最低限度的查詢數目，則會在啟動叢集後的數小時內套用最佳化。

如果 Amazon Redshift 判斷分佈索引鍵可改善查詢的效能，則分佈樣式為 AUTO 的資料表可以將其分佈樣式變更為 KEY。

主題

啟用自動資料表最佳化

根據預設，未明確定義排序索引鍵或分佈索引鍵而建立的資料表會設定為 AUTO。在建立資料表時，您也可以手動明確設定排序或分佈索引鍵。如果您設定排序或分佈索引鍵，則資料表不會自動受到管理。

若要對現有資料表啟用自動最佳化，請使用 ALTER 陳述式選項將資料表變更為 AUTO。您可以選擇為排序索引鍵定義自動化，但不能定義分佈索引鍵的自動化 (反之亦然)。如果您執行 ALTER 陳述式，將資料表轉換為自動化資料表，則會保留現有的排序索引鍵和分佈樣式。

```
ALTER TABLE table_name ALTER SORTKEY AUTO;
```

```
ALTER TABLE table_name ALTER DISTSTYLE AUTO;
```

如需詳細資訊，請參閱 ALTER TABLE。

一開始，資料表沒有分佈索引鍵或排序索引鍵。分佈樣式會設定為 EVEN 或 ALL，取決於資料表大小。隨著資料表的大小不斷增加，Amazon Redshift 會套用最佳分佈索引鍵和排序索引鍵。最佳化會在執行最少查詢數目後的小時內套用。在決定排序索引鍵最佳化時，Amazon Redshift 會嘗試最佳化資料表掃描期間從磁碟讀取的資料區塊。在決定分佈樣式最佳化時，Amazon Redshift 會嘗試最佳化叢集節點之間傳輸的位元組數。

從資料表中移除自動資料表最佳化

您可以從自動最佳化中移除資料表。從自動化中移除資料表需要選取排序索引鍵或分佈樣式。若要變更分佈樣式，請指定特定的分佈樣式。

```
ALTER TABLE table_name ALTER DISTSTYLE EVEN;
```

```
ALTER TABLE table_name ALTER DISTSTYLE ALL;
```

```
ALTER TABLE table_name ALTER DISTSTYLE KEY DISTKEY c1;
```

若要變更排序索引鍵，您可以定義排序索引鍵或選擇「無」。

```
ALTER TABLE table_name ALTER SORTKEY(c1, c2);
```

```
ALTER TABLE table_name ALTER SORTKEY NONE;
```

監控自動資料表最佳化的動作

系統檢視 SVV_ALTER_TABLE_RECOMMENDATIONS 會記錄資料表目前的 Amazon Redshift Advisor 建議。此檢視會顯示所有資料表的建議，包括已針對或未針對自動最佳化定義的資料表。

若要檢視資料表是否已定義為自動最佳化，請查詢系統檢視 SVV_TABLE_INFO。項目只會針對目前工作階段資料庫中可見的資料表而顯示。建議會在叢集建立後的幾小時內開始插入檢視 (每天兩次)。建議

可用之後，就會在一小時內開始執行。在套用建議之後 (無論是由 Amazon Redshift 或您執行)，該建議將不再出現在檢視中。

系統檢視 SVL_AUTO_WORKER_ACTION 會顯示 Amazon Redshift 採取的所有動作的稽核日誌，以及資料表的先前狀態。

系統檢視 SVV_TABLE_INFO 會列出系統中的所有資料表，以及用來指出資料表的排序索引鍵和分佈樣式是否設定為 AUTO 的資料欄。

如需這些系統檢視的相關資訊，請參閱 系統監控 (僅限已佈建)。

使用資料欄壓縮

壓縮是欄層級操作，可降低資料儲存時的大小。壓縮可節省儲存空間，並降低從儲存體讀取的資料大小，這樣會減少磁碟 I/O 數量，因而改善查詢效能。

ENCODE AUTO 是資料表的預設值。當資料表設定為 ENCODE AUTO 時，Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。如需詳細資訊，請參閱 CREATE TABLE 及 ALTER TABLE。

不過，如果您為資料表中的任何資料欄指定壓縮編碼，資料表就不會再設定為 ENCODE AUTO。Amazon Redshift 不再自動管理資料表中所有資料欄的壓縮編碼。

建立資料表時，您可以手動將壓縮類型或編碼套用至資料表中的資料欄。或者，您可以使用 COPY 命令自動分析和套用壓縮。如需詳細資訊，請參閱 讓 COPY 選擇壓縮編碼。如要套用自動壓縮的詳細資訊，請參閱利用自動壓縮載入資料表。

> **Note**
>
> 我們強烈建議使用 COPY 命令來套用自動壓縮。

如果新資料表與另一個資料表共用相同的資料特性，您可以選擇手動套用壓縮編碼。或者，如果您在測試過程中發現自動壓縮過程中套用的壓縮編碼不適合您的資料，則可能會這樣做。如果您選擇手動套用壓縮編碼，則可以針對已填入的資料表執行 ANALYZE COMPRESSION 命令，並使用結果來選擇壓縮編碼。

若要手動套用壓縮，您可以將個別直欄的壓縮編碼指定為 CREATE TABLE 陳述式的一部分。語法如下。

```
CREATE TABLE table_name (column_name
```

```
data_type ENCODE encoding-type)[, ...]
```

其中 encoding-type 是取自於下節中的關鍵字資料表。

例如，下列陳述式會建立兩欄資料表 PRODUCT。將資料載入至資料表時，不會使用位元組字典編碼 (BYTEDICT) 來壓縮 PRODUCT_ID 資料欄，但會壓縮 PRODUCT_NAME 資料欄。

```
create table product(
product_id int encode raw,
product_name char(20) encode bytedict);
```

在使用 ALTER TABLE 陳述式，將資料欄新增至資料表時，您可以指定該資料欄的編碼。

```
ALTER TABLE table-name ADD [ COLUMN ] column_name column_type ENCODE encoding-type
```

主題

壓縮編碼

壓縮編碼會指定在資料列新增至資料表時，套用至一欄資料值的壓縮類型。

ENCODE AUTO 是資料表的預設值。當資料表設定為 ENCODE AUTO 時，Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。如需詳細資訊，請參閱 CREATE TABLE 及 ALTER TABLE。

不過，如果您為資料表中的任何資料欄指定壓縮編碼，資料表就不會再設定為 ENCODE AUTO。Amazon Redshift 不再自動管理資料表中所有資料欄的壓縮編碼。

當您使用 CREATE TABLE 時，當您為資料表中的任何資料欄指定壓縮編碼時，會停用 ENCODE AUTO。如果停用 ENCODE AUTO，Amazon Redshift 會自動將壓縮編碼指派給您未指定 ENCODE 類型的資料欄，如下所示：

- 定義為排序索引鍵的資料欄會有指派的 RAW 壓縮。
- 定義為 BOOLEAN、REAL 或 DOUBLE PRECISION 資料類型的資料欄會有指派的 RAW 壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIMESTAMP 或 TIMESTAMPTZ 資料類型的資料欄會有指派的 AZ64 壓縮。

- 定義為 CHAR 或 VARCHAR 資料類型的資料欄會有指派的 LZO 壓縮。

您可以使用 ALTER TABLE 在建立資料表之後變更資料表的編碼。如果您使用 ALTER 資料表停用 ENCODE AUTO 功能，Amazon Redshift 將不再自動管理資料欄的壓縮編碼。所有資料欄都會保留您停用 ENCODE 時的壓縮編碼類型，直到您將其變更或再次啟用 ENCODE AUTO 為止。

下表識別支援的壓縮編碼和支援編碼的資料類型。

編碼類型	CREATE TABLE 和 ALTER TABLE 中的關鍵字	資料類型
Raw (無壓縮)	RAW	全部
AZ64	AZ64	SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIMESTAMP、TIMESTAMPTZ
Byte dictionary	BYTEDICT	SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE PRECISION、CHAR、VARCHAR、DATE、TIMESTAMP、TIMESTAMPTZ
Delta	DELTA DELTA32K	SMALLINT、INT、BIGINT、DATE、TIMESTAMP、DECIMAL INT、BIGINT、DATE、TIMESTAMP、DECIMAL
LZO	LZO	SMALLINT、INTEGER、BIGINT、DECIMAL、CHAR、VARCHAR、DATE、TIMESTAMP、TIMESTAMPTZ、SUPER
Mostlyn	MOSTLY8 MOSTLY16	SMALLINT、INT、BIGINT、DECIMAL INT、BIGINT、DECIMAL

編碼類型	CREATE TABLE 和 ALTER TABLE 中的關鍵字	資料類型
	MOSTLY32	BIGINT、DECIMAL
Run-length	RUNLENGTH	SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE PRECISION、BOOLEAN、CHAR、VARCHAR、DATE、TIMESTAMP、TIMESTAMPTZ
文字	TEXT255 TEXT32K	僅限 VARCHAR 僅限 VARCHAR
Zstandard	ZSTD	SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE PRECISION、BOOLEAN、CHAR、VARCHAR、DATE、TIMESTAMP、TIMESTAMPTZ、SUPER

Raw 編碼

Raw 編碼是下列資料欄的預設編碼：指定為排序索引鍵的資料欄，以及定義為 BOOLEAN、REAL 或 DOUBLE PRECISION 資料類型的資料欄。系統使用 Raw 編碼，以原始、未壓縮格式來儲存資料。

AZ64 編碼

AZ64 是 Amazon 設計的專屬壓縮編碼，旨在實現高壓縮比並改善查詢處理能力。AZ64 演算法的核心是壓縮更小的一組資料值，並使用單指令、多資料 (SIMD) 指令進行並行處理。使用 AZ64 為數字、日期及時間資料類型節省大量儲存空間並實現高效能。

當搭配下列資料類型使用 CREATE TABLE 及 ALTER TABLE 陳述式，來定義資料欄時，您可以使用 AZ64 做為壓縮編碼：

- SMALLINT
- INTEGER

- BIGINT
- DECIMAL
- DATE
- TIMESTAMP
- TIMESTAMPTZ

Byte-Dictionary 編碼

在 byte dictionary 編碼中，唯一值的個別字典是針對磁碟上資料欄值的每個區塊而建立的。(Amazon Redshift 磁碟會佔用 1 MB。) 字典最多可包含 256 個一元位元組值，其會以索引形式儲存至原始資料值。如果在單一區塊中儲存超過 256 個值，則額外的值會以原始、未壓縮格式寫入至區塊。對於每個磁碟區塊都會重複此程序。

這種編碼對低基數字串資料欄非常有效。當資料欄的資料網域少於 256 個唯一值時，這是最佳的編碼。

對於字串資料類型 (CHAR 和 VARCHAR) 使用 BYTEDICT 編碼的資料欄，Amazon Redshift 會執行向量化掃描和述詞評估，直接對壓縮資料進行操作。這些掃描會使用硬體專屬的單一指示和多重資料 (SIMD) 指示進行平行處理。這會顯著加快字串資料欄的掃描速度。如果 CHAR/VARCHAR 資料欄保存長字元字串，則 Byte-dictionary 編碼特別節省空間。

假設資料表具有資料類型為 CHAR(30) 的 COUNTRY 資料欄。載入資料時，Amazon Redshift 會建立字典。並在 COUNTRY 資料欄填入索引值。字典包含已建立索引的唯一值，而且資料表本身只包含對應值的一位元組下標。

> **Note**
>
> 會儲存固定長度字元欄的多餘空格。因此，在 CHAR(30) 資料欄中，當您使用 byte-dictionary 編碼時，每個對應值都會節省 29 個位元組的儲存空間。

下表代表 COUNTRY 資料欄的字典。

唯一資料值	字典索引	大小 (固定長度，每個值 30 個位元組)
England	0	30

唯一資料值	字典索引	大小 (固定長度，每個值 30 個位元組)
United States of America	1	30
Venezuela	2	30
Sri Lanka	3	30
Argentina	4	30
Japan	5	30
合計		180

下表代表 COUNTRY 資料欄中的值。

原始資料值	原始大小 (固定長度，每個值 30 個位元組)	壓縮值 (索引)	新大小 (位元組)
England	30	0	1
England	30	0	1
United States of America	30	1	1
United States of America	30	1	1
Venezuela	30	2	1
Sri Lanka	30	3	1
Argentina	30	4	1
Japan	30	5	1
Sri Lanka	30	3	1

原始資料值	原始大小 (固定長度，每個值 30 個位元組)	壓縮值 (索引)	新大小 (位元組)
Argentina	30	4	1
合計	300		10

此範例中的合計壓縮大小計算如下：6 個不同項目儲存在字典 (6 * 30 = 180)，而且資料表包含 10 個 1 位元組壓縮值，合計 190 個位元組。

Delta 編碼

Delta 編碼對日期時間欄很有用。

Delta 藉由記錄資料欄中彼此跟隨的值之間的差異來壓縮資料。此差異會記錄在磁碟上資料欄值之每個區塊的個別字典中。(Amazon Redshift 磁碟會佔用 1 MB。) 例如，假設資料欄包含 10 個整數，順序從 1 到 10。第一個會儲存為 4 位元組整數 (加上 1 位元組旗標)。接下來的九個會分別儲存為值為 1 的位元組，表示比前一個值多 1。

Delta 編碼附有兩個變異：

- DELTA 會將差異記錄為 1 位元組值 (8 位元組整數)
- DELTA32K 會將差異記錄為 2 位元組值 (16 位元組整數)

如果資料欄中的大多數值可以透過使用單一位元組進行壓縮，則 1 位元組的變化非常有效。但是，如果差異較大，則此編碼 (在最壞的情況下) 會比儲存未壓縮資料的效果差。類似邏輯適用於 16 位元版本。

如果兩個值之間的差異超過 1 位元組範圍 (DELTA) 或 2 位元組範圍 (DELTA32K)，則會儲存完整原始值，前面為 1 位元組旗標。1 位元組範圍從 -127 到 127，而 2 位元組範圍從 -32K 到 32K。

下表顯示數值欄的 Delta 編碼如何運作。

原始資料值	原始大小 (位元組)	差異 (delta)	壓縮值	壓縮大小 (位元組)
1	4		1	1+4 (旗標 + 實際值)

原始資料值	原始大小 (位元組)	差異 (delta)	壓縮值	壓縮大小 (位元組)
5	4	4	4	1
50	4	45	45	1
200	4	150	150	1+4 (旗標 + 實際值)
185	4	-15	-15	1
220	4	35	35	1
221	4	1	1	1
合計	28			15

LZO 編碼

LZO 壓縮編碼提供非常高的壓縮率和良好的效能。LZO 編碼特別適用於儲存很長的字元字串的 CHAR 和 VARCHAR 資料欄。它們特別適用於自由格式文字，例如產品說明、使用者註解或 JSON 字串。

Mostly 編碼

當資料欄的資料類型大於大部份儲存值所需的資料類型時，Mostly 編碼很有用。您可以對此類型的資料欄指定 mostly 編碼，將資料欄中的大部分值壓縮為更小的標準儲存空間大小。剩下無法壓縮的值會以其原始格式儲存。例如，您可以將 16 位元資料欄 (例如 INT2 資料欄) 壓縮為 8 位元儲存空間。

通常，mostly 編碼會使用下列資料類型：

- SMALLINT/INT2 (16 位元)
- INTEGER/INT (32 位元)
- BIGINT/INT8 (64 位元)
- DECIMAL/NUMERIC (64 位元)

選擇 mostly 編碼的適當變異來符合資料欄之資料類型的大小。例如，將 MOSTLY8 套用至定義為 16 位元整數資料欄的資料欄。不允許將 MOSTLY16 套用至具有 16 位元資料類型的資料欄，也不允許將 MOSTLY32 套用至具有 32 位元資料類型的資料欄。

當資料欄中有相當多的值無法壓縮時，Mostly 編碼可能比無壓縮更沒效率。在將其中一種編碼套用到資料欄之前，請執行檢查。您現在將載入的大部分值 (也可能在未來載入) 應納入下表中顯示的範圍。

編碼	壓縮儲存空間大小	可以壓縮的值範圍 (範圍外的值會以原始形式儲存)
MOSTLY8	1 位元組 (8 位元)	-128 到 127
MOSTLY16	2 位元組 (16 位元)	-32768 到 32767
MOSTLY32	4 位元組 (32 位元)	-2147483648 到 +2147483647

> **Note**
>
> 對於小數，忽略小數點以判斷值是否納入範圍。例如，1,234.56 會視為 123,456，且可在 MOSTLY32 資料欄中壓縮。

例如，VENUE 資料表中的 VENUEID 資料欄會定義為原始整數資料欄，這表示其值佔用 4 位元組的儲存空間。不過，此欄的值目前範圍是 **0** 到 **309**。因此，針對 VENUEID 使用 MOSTLY16 編碼來重建和重新載入此資料表，會將該資料欄中每個值的儲存空間減少至 2 個位元組。

如果另一個資料表中參考的 VENUEID 值大部分在範圍 0 到 127，則將該外部索引鍵資料欄編碼為 MOSTLY8 可能就有意義。在做出選擇之前，請針對參考資料表資料執行數個查詢，以了解值是否大部分都落入 8 位元、16 位元或 32 位元範圍。

下表顯示在使用 MOSTLY8、MOSTLY16 和 MOSTLY32 編碼時特定數值的壓縮大小：

原始值	原始 INT 或 BIGINT 大小 (位元組)	MOSTLY8 壓縮大小 (位元組)	MOSTLY16 壓縮大小 (位元組)	MOSTLY32 壓縮大小 (位元組)
1	4	1	2	4
10	4	1	2	4
100	4	1	2	4

原始值	原始 INT 或 BIGINT 大小 (位元組)	MOSTLY8 壓縮大小 (位元組)	MOSTLY16 壓縮大小 (位元組)	MOSTLY32 壓縮大小 (位元組)
1000	4	與原始資料大小相同	2	4
10000	4		2	4
20000	4		2	4
40000	8		與原始資料大小相同	4
100000	8			4
2000000000	8			4

執行長度編碼

執行長度編碼會將連續重複的值取代為由值和連續出現次數 (執行長度) 組成的符記。唯一值的個別字典是針對磁碟上資料欄值的每個區塊而建立的。(Amazon Redshift 磁碟會佔用 1 MB。) 此編碼最適合於資料值通常連續重複的資料表，例如，依那些值排序資料表時。

例如，假設大型維度資料表中的資料欄具有可預測的較小領域，例如 COLOR 資料欄的可能值少於 10 個。這些值可能會落在整個資料表中的長序列中，即使資料未排序也是如此。

不建議在指定為排序索引鍵的任何資料欄上套用執行長度編碼。當區塊包含類似的列數時，限制範圍的掃描執行效果會更好。如果排序索引鍵欄位的壓縮比相同查詢中的其他欄位高出許多，限制範圍的掃描執行效果可能較差。

下表使用 COLOR 資料欄範例來顯示執行長度編碼如何運作。

原始資料值	原始大小 (位元組)	壓縮值 (符記)	壓縮大小 (位元組)
Blue	4	{2,藍色}	5
Blue	4		0
Green	5	{3,綠色}	6
Green	5		0

原始資料值	原始大小 (位元組)	壓縮值 (符記)	壓縮大小 (位元組)
Green	5		0
Blue	4	{1,Blue}	5
Yellow	6	{4,Yellow}	7
Yellow	6		0
Yellow	6		0
Yellow	6		0
合計	51		23

Text255 和 Text32k 編碼

Text255 和 text32k 編碼有助於相同單字經常重複出現的壓縮 VARCHAR 資料欄。唯一單字的個別字典是針對磁碟上資料欄值的每個區塊而建立的。(Amazon Redshift 磁碟會佔用 1 MB。) 字典包含資料欄中前 245 個唯一單字。那些單字會在磁碟上被代表 245 個值之一的一位元組索引值所取代，而且未在字典中表示的任何單字會以未壓縮形式儲存。對於每個 1 MB 磁碟區塊都會重複此程序。如果建立索引的單字經常出現在資料欄中，則資料欄將產生高壓縮率。

對於 text32k 編碼，原則相同，但每個區塊的字典不會擷取特定數目的單字。反之，字典會為其找到的每個唯一單字建立索引，直到結合的項目達到 32K 長度，減去一些額外負荷。索引值是以兩個位元組儲存。

例如，考慮 VENUE 資料表中的 VENUENAME 資料欄。**Arena**、**Center** 和 **Theatre** 等單字反覆出現在這個欄中，而且若套用 text255 壓縮，則很可能出現在每個區塊中的前 245 個單字之中。如果是這樣，則此資料欄將受益於壓縮。因為每次那些單字出現，它們將只佔用 1 個位元組的儲存空間 (而不是分別佔用 5、6 或 7 個位元組)。

Zstandard 編碼

對於各種資料集，Zstandard (ZSTD) 編碼提供高壓縮率和極佳效能。ZSTD 編碼特別適用於儲存廣泛長短字串的 CHAR 和 VARCHAR 資料欄，特別是自由格式的文字，例如產品描述、使用者意見、日誌和 JSON 字串。其中某些演算法 (例如 Delta 編碼或 Mostly 編碼) 可以潛在比無壓縮使用更多的儲存空間，而 ZSTD 不太可能增加磁碟用量。

ZSTD 支援 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE PRECISION、BOOLEAN、CHAR、VARCHAR、DATE、TIMESTAMP 和 TIMESTAMPTZ 等資料類型。

測試壓縮編碼

如果您決定手動指定資料欄編碼，可能想要使用您的資料測試不同編碼。

> **Note**
>
> 建議您儘可能使用 COPY 命令來載入資料，並允許 COPY 命令根據您的資料選擇最佳的編碼。或者，您可以使用 ANALYZE COMPRESSION 命令，來檢視對現有資料建議的編碼。如要套用自動壓縮的詳細資訊，請參閱利用自動壓縮載入資料表。

若要執行有意義的資料壓縮測試，您必須有大量資料列。在此範例中，我們將使用從兩個資料表 VENUE和 LISTING 中選取的陳述式，來建立資料表並插入資料列。我們省略了通常會連接兩個表的 WHERE 子句。結果是 VENUE 資料表中的每個資料列都會連接至 LISTING 資料表中的所有資料列，總數超出 3 千 2 百萬個資料列。這稱為笛卡爾連接，通常不建議使用。但是，為此，這是建立許多資料列的便捷方法。如果您的現有資料表具有您要測試的資料，則可略過此步驟。

在我們有一個包含範例資料的資料表後，我們會建立一個包含七個資料欄的資料表。每個資料欄都有不同的壓縮編碼：raw、bytedict、lzo、run length、text255、text32k 和 zstd。我們會執行從第一個資料表中選取資料的 INSERT 命令，在每一個資料欄中填入完全相同的資料。

若要測試壓縮編碼，請執行以下動作：

1. (選用) 首先，使用笛卡兒連接來建立一個具有大量資料列的資料表。如果您想要測試現有資料表，請略過此步驟。

```
create table cartesian_venue(
venueid smallint not null distkey sortkey,
venuename varchar(100),
venuecity varchar(30),
venuestate char(2),
venueseats integer);

insert into cartesian_venue
select venueid, venuename, venuecity, venuestate, venueseats
from venue, listing;
```

2. 接著，以您想要比較的編碼建立資料表。

```
create table encodingvenue (
venueraw varchar(100) encode raw,
venuebytedict varchar(100) encode bytedict,
venuelzo varchar(100) encode lzo,
venuerunlength varchar(100) encode runlength,
venuetext255 varchar(100) encode text255,
venuetext32k varchar(100) encode text32k,
venuezstd varchar(100) encode zstd);
```

3. 使用 INSERT 陳述式與 SELECT 子句搭配，將相同資料插入至所有資料欄。

```
insert into encodingvenue
select venuename as venueraw, venuename as venuebytedict, venuename as venuelzo,
 venuename as venuerunlength, venuename as  venuetext32k, venuename as  venuetext255,
 venuename as venuezstd
from cartesian_venue;
```

4. 驗證新資料表中的資料列數量。

```
select count(*) from encodingvenue

  count
----------
 38884394
(1 row)
```

5. 查詢 STV_BLOCKLIST 系統資料表，以比較每個資料欄使用的 1 MB 磁碟區塊數。

   MAX 彙總函數會傳回每個欄的最高區塊數。STV_BLOCKLIST 資料表包含三個系統產生之資料欄的詳細資訊。此範例在 WHERE 子句中使用 `col < 6` 來排除系統產生的資料欄。

```
select col, max(blocknum)
from stv_blocklist b, stv_tbl_perm p
where (b.tbl=p.id) and name ='encodingvenue'
and col < 7
group by name, col
order by col;
```

   此查詢會傳回下列結果。資料欄是從零開始編號。根據您設定叢集的方式，您的結果可能具有不同號碼，但相對大小應該類似。您可以看出第二欄的 BYTEDICT 編碼產生此資料集的最佳結果。這種

方法具有比 20:1 更好的壓縮比。LZO 和 ZSTD 編碼也會產生絕佳的結果。當然，不同資料集會產生不同結果。當資料欄包含更長的文字字串時，LZO 通常會產生最好的壓縮結果。

```
 col | max
-----+-----
   0 | 203
   1 |  10
   2 |  22
   3 | 204
   4 |  56
   5 |  72
   6 |  20
(7 rows)
```

如果您有現有資料表中的資料，則您可以使用 ANALYZE COMPRESSION 命令，來檢視對資料表建議的編碼。例如，下列範例顯示對 VENUE 資料表 (CARTESIAN_VENUE) 的副本建議的編碼，而此資料表包含 3 千 8 百萬個資料列。請注意，ANALYZE COMPRESSION 建議對為 VENUENAME 資料欄使用 LZO 編碼。ANALYZE COMPRESSION 根據多個因素 (包含減少百分比) 選擇最佳的壓縮。在此特定的情況下，BYTEDICT 提供更好的壓縮，但 LZO 也會產生大於 90% 的壓縮。

```
analyze compression cartesian_venue;

Table          | Column     | Encoding | Est_reduction_pct
---------------+------------+----------+------------------
reallybigvenue | venueid    | lzo      | 97.54
reallybigvenue | venuename  | lzo      | 91.71
reallybigvenue | venuecity  | lzo      | 96.01
reallybigvenue | venuestate | lzo      | 97.68
reallybigvenue | venueseats | lzo      | 98.21
```

範例：選擇 CUSTOMER 資料表的壓縮編碼

下列陳述式會建立一個 CUSTOMER 資料表，其資料欄具有各種資料類型。此 CREATE TABLE 陳述式會為這些資料欄顯示壓縮編碼的許多可能組合的其中一個。

```
create table customer(
custkey int encode delta,
custname varchar(30) encode raw,
gender varchar(7) encode text255,
address varchar(200) encode text255,
```

```
city varchar(30) encode text255,
state char(2) encode raw,
zipcode char(5) encode bytedict,
start_date date encode delta32k);
```

下表顯示已針對 CUSTOMER 資料表選擇的資料欄，並提供選擇的說明：

資料行	資料類型	編碼	說明
CUSTKEY	int	delta	CUSTKEY 包含唯一連續的整數值。因為差異將為一個位元組，所以 DELTA 是個好選擇。
CUSTNAME	varchar(30)	raw	CUSTNAME 具有重複值只有幾個的大型網域。任何壓縮編碼通常可能成效不佳。
GENDER	varchar(7)	text255	GENDER 是重複值很多的超小型網域。Text255 很適合用於相同單字重複出現的 VARCHAR 資料欄。
ADDRESS	varchar(200)	text255	ADDRESS 是大型網域，但包含許多重複單字，例如 Street Avenue、North、South 等等。Text 255 和 text 32k 編碼有助於相同單字重複出現的壓縮 VARCHAR 資料欄。資料欄長度很短，因

資料行	資料類型	編碼	說明
			此 text255 是個好選擇。
CITY	varchar(30)	text255	CITY 是只有一些重複值的大型網域。特定城市名稱比其他城市名稱更常使用。Text 255 是個好選擇，原因與 ADDRESS 相同。
STATE	char(2)	raw	在美國，STATE 是 50 個兩字元值的精確網域。Bytedict 編碼將產生一些壓縮，但是因為資料欄大小只是兩個字元，所以壓縮可能不值得解除壓縮資料的額外負荷。
ZIPCODE	char(5)	bytedict	ZIPCODE 是少於 50,000 個唯一值的已知網域。特定郵遞區號比其他郵遞區號更常出現。當資料欄包含有限數目的唯一值時，Bytedict 編碼很有效。
START_DATE	date	delta32k	Delta 編碼對日期時間資料欄很有用，尤其是在資料列依日期順序載入時。

使用資料分佈樣式

當您將資料載入資料表時，Amazon Redshift 會根據資料表的分佈樣式，將資料表的資料列分佈至每一個運算節點。當您執行查詢時，查詢最佳化工具會視需要將資料列重新配送至運算節點，以執行任何聯結與彙總。選擇資料表配送樣式的目的是，藉由在執行查詢之前，將資料配置至必要的位置，以降低重新配送步驟所帶來的影響。

> **Note**
>
> 本節將向您介紹 Amazon Redshift 資料庫中資料分佈的原則。建議您使用 `DISTSTYLE AUTO` 建立資料表。如果您這樣做，Amazon Redshift 會使用自動資料表最佳化來選擇資料分發樣式。如需詳細資訊，請參閱 使用自動資料表最佳化。本節的其餘部分提供有關分佈樣式的詳細資料。

主題

資料分佈概念

Amazon Redshift 的一些資料分佈概念如下。

節點和配量

Amazon Redshift 叢集是一組節點。叢集中的每個節點都有自己的作業系統、專用的記憶體，以及專用的磁碟儲存空間。一個節點是領導者節點，其會管理運算節點的資料分佈和查詢處理任務。運算節點提供執行這些工作的資源。

運算節點的磁碟儲存空間會分成一些配量。每一節點的配量數目取決於叢集的節點大小。節點全部會參與在執行的平行查詢中，以處理盡可能平均地分佈於配量的資料。如需每個節點大小有多少配量的相關資訊，請參閱《Amazon Redshift 管理指南》中的關於叢集和節點。

資料重新分佈

當您將資料載入資料表時，Amazon Redshift 會根據資料表的分佈樣式，將資料表的資料列分佈至每一個節點配量。做為查詢計畫的一部分，最佳化器會決定資料區塊需要位於何處，才能最佳地執行查詢。然後，系統會在查詢執行時實際移動或重新分配資料。重新分佈可能需要將特定資料列傳送至節點，以將整個資料表聯結或廣播至所有節點。

資料重新分佈可以佔查詢計劃成本的很大一部分，並且它產生的網絡流量可能會影響其他資料庫操作，並降低整體系統效能。在某種程度上，您可以預期最初定位資料的最佳位置，將資料重新分佈的影響降至最低。

資料分佈目標

當您將資料載入資料表時，Amazon Redshift 會根據您建立資料表時選擇的分佈樣式，將資料表的資料列分佈至運算節點及配量。資料分佈具有兩個主要目標：

- 在叢集的節點之間平均分配工作負載。分佈不平均 (或資料分佈扭曲) 會迫使某些節點比其他節點更忙碌工作，導致查詢效能下降。
- 為了盡量減少查詢執行時的資料移動。如果參與聯結或彙總的資料列已在其聯結資料列位於其他資料表的叢集上共置，則最佳化器不需要在查詢執行期間重新分佈儘可能多的資料。

您為資料庫選擇的分佈策略對查詢效能、儲存空間需求、資料載入和維護具有重要影響。藉由為每個資料表選擇最好的分佈樣式，您可以平衡資料分佈，並大幅改善整體系統效能。

分佈樣式

建立表格時，您可以指定下列其中一種分佈樣式：「自動」、「偶數」、「KEY」或「全部」。

如果您不指定分佈樣式，Amazon Redshift 會使用 AUTO 分佈。

AUTO 分佈

使用 AUTO 分佈時，Amazon Redshift 會根據資料表資料的大小來指派最佳分佈樣式。例如，如果指定 AUTO 分佈樣式，Amazon Redshift 一開始會將 ALL 分佈樣式指派給小型資料表。當資料表變大時，Amazon Redshift 可能會將分佈樣式變更為 KEY，並選擇主索引鍵 (或複合主鍵的資料欄) 作為分佈索引鍵。如果資料表變大，並且沒有任何資料欄適合作為分佈索引鍵，則 Amazon Redshift 會將分佈樣式變更為 EVEN。分佈樣式的變更會發生在背景中，對使用者查詢的影響最小。

若要檢視 Amazon Redshift 自動執行的動作來修改資料表分佈索引鍵，請參閱 SVL_AUTO_WORKER_ACTION。若要檢視有關修改資料表分佈索引鍵的目前建議，請參閱 SVV_ALTER_TABLE_RECOMMENDATIONS。

若要檢視套用至資料表的分佈樣式，請查詢 PG_CLASS_INFO 系統目錄檢視。如需詳細資訊，請參閱檢視分佈樣式。如果您沒有以 CREATE TABLE 陳述式指定分佈樣式，Amazon Redshift 會套用 AUTO 分佈。

EVEN 分佈

領導者節點會以循環模式在配量之間分佈資料列，而不考慮任何特定資料欄的值。當資料表不參與聯結時，EVEN 分佈是適當做法。當 KEY 分佈和 ALL 分佈之間沒有明確的選擇時，這也是合適的。

KEY 分佈

資料列根據一個欄的值來分佈。領導節點會將相符的值放在相同的節點配量。如果您根據聯結索引鍵來分佈一對資料表，領導者節點會根據聯結欄的值將資料列共置在配量。這樣，來自共同資料欄的比對值會實際儲存在一起。

ALL 分佈

整個資料表的副本分佈至每個節點。EVEN 分佈或 KEY 分佈只會將資料表的一部分資料列放在每個節點上，而 ALL 分佈可確保資料表參與的每個聯絡都共置每個資料列。

ALL 分佈會增加叢集中節點數所需的儲存空間，因此需要更多時間來載入資料、更新資料，或將資料插入至多個資料表。ALL 分佈僅適用於移動速度相當緩慢的資料表，亦即，不經常或廣泛更新的資料表。因為在查詢期間重新分配小型資料表的成本很低，所以將小型維度資料表定義為 DISTSTYLE ALL 沒有顯著的優勢。

> **Note**
>
> 指定資料欄的分佈樣式之後，Amazon Redshift 就會在叢集層級上處理資料分佈。Amazon Redshift 不需要或不支援在資料庫物件內分割資料的概念。您不需要建立資料表空間或定義資料表的分割方案。

在某些情況下，您可以在建立資料表的分佈樣式之後，變更該分佈樣式。如需詳細資訊，請參閱 ALTER TABLE。對於建立資料表分佈樣式之後無法加以變更的情況，您可以重新建立資料表，並以深層複製填入新資料表。如需更多資訊，請參閱執行深層複製

檢視分佈樣式

若要檢視資料表的分佈樣式，請查詢 PG_CLASS_INFO 檢視或 SVV_TABLE_INFO 檢視。

PG_CLASS_INFO 中的 RELEFFECTIVEDISTSTYLE 欄指出資料表的目前分佈樣式。如果資料表使用自動分佈，則 RELEFFECTIVEDISTSTYLE 為 10、11 或 12，這指出有效分佈樣式為 AUTO (ALL)、AUTO (EVEN) 或 AUTO (KEY)。如果資料表使用自動分布，則分佈樣式一開始可能顯示 AUTO (ALL)，然後在資料表變大時，變更為 AUTO (EVEN) 或 AUTO (KEY)。

下表提供 RELEFFECTIVEDISTSTYLE 欄中每個值的分佈樣式：

RELEFFECTIVEDISTSTYLE	目前分佈樣式
0	EVEN
1	KEY
8	ALL
10	AUTO (ALL)
11	AUTO (EVEN)
12	AUTO (KEY)

SVV_TABLE_INFO 中的 DISTSTYLE 欄指出資料表的目前分佈樣式。如果資料表使用自動分佈，則 DISTSTYLE 為 AUTO (ALL)、AUTO (EVEN) 或 AUTO (KEY)。

下列範例使用三個分佈樣式和自動分佈建立四個資料表，然後查詢 SVV_TABLE_INFO 來檢視分佈樣式。

```
create table public.dist_key (col1 int)
diststyle key distkey (col1);

insert into public.dist_key values (1);

create table public.dist_even (col1 int)
diststyle even;
```

```
insert into public.dist_even values (1);

create table public.dist_all (col1 int)
diststyle all;

insert into public.dist_all values (1);

create table public.dist_auto (col1 int);

insert into public.dist_auto values (1);

select "schema", "table", diststyle from SVV_TABLE_INFO
where "table" like 'dist%';

        schema   |    table        | diststyle
     ------------+-----------------+------------
      public     | dist_key        | KEY(col1)
      public     | dist_even       | EVEN
      public     | dist_all        | ALL
      public     | dist_auto       | AUTO(ALL)
```

評估查詢模式

選擇分佈樣式只是資料庫設計的一個環節。請在整個系統的內容中考慮分佈樣式，同時使用其他重要因素來平均分佈，例如叢集大小、壓縮編碼方法、排序索引鍵，以及資料表限制。

利用儘可能接近真實資料的資料來測試您的系統。

若要為分佈樣式做出好選擇，您必須了解 Amazon Redshift 應用程式的查詢模式。識別系統中成本最高的查詢，並根據那些查詢的需求來設計初始的資料庫。決定查詢總成本的因素包括查詢執行所需的時間長度，以及耗用多少運算資源。決定查詢成本的其他因素是執行的頻率，以及對其他查詢和資料庫作業的干擾程度。

識別成本最高查詢使用的資料表，並評估其在查詢執行期中的角色。請考慮聯結和彙總資料表的方式。

使用本節中的準則來選擇每個資料表的分佈樣式。當您這樣做時，請建立資料表，並將儘可能接近真實資料的資料載入其中。然後測試資料表是否有您希望使用的查詢類型。您可以評估查詢解釋計畫來識別調校機會。比較載入時間、儲存空間和查詢執行期，以便平衡系統的整體需求。

指定分佈樣式

本節中指定分佈樣式的考量和建議會使用星狀結構描述做為範例。您的資料庫設計可能根據星狀結構描述、某個星狀結構描述變體，或完全不同的結構描述。Amazon Redshift 會設計為可搭配您選擇的任何結構描述設計而有效運作。本節中的原理適用於任何設計結構描述。

1. 指定您的所有資料表的主索引鍵和外部索引鍵。

   Amazon Redshift 不會強制執行主索引鍵和外部索引鍵限制，但查詢最佳化器會在產生查詢計劃時使用這些限制。如果您設定主索引鍵和外部索引鍵，則您的應用程式必須維護索引鍵的有效性。
2. 將事實資料表與其最大維度資料表配送至它們的通用欄位。

   根據參與最常用聯結的資料集大小，而不只是資料表的大小來選擇最大維度。如果通常使用 WHERE 子句來篩選資料表，則只有其資料列的一部分參與聯結。這類資料表對重新分佈比提供更多資料的更小資料表具有更少的影響。將維度資料表的主索引鍵與事實資料表的對應外部索引鍵指定為 DISTKEY。如果多個資料表使用相同的分佈索引鍵，它們也將與事實資料表共置。您的事實資料表只能有一個分佈索引鍵。聯結其他索引鍵的任何資料表都不會與事實資料表共置。
3. 指定其他維度資料表的分佈索引鍵。

   根據其主索引鍵或外部索引鍵來分佈資料表，取決於它們最常與其他資料表聯結的方式。
4. 評估是否將一些維度資料表變更為使用 ALL 分佈。

   如果維度資料表無法與事實資料表或其他重要的聯結資料表共置，您可以藉由將整個資料表發佈至所有節點，以大幅提升查詢效能。使用 ALL 分佈會增加儲存空間要求，並增加載入時間與維護操作，因此在選擇 ALL 分佈之前，您應權衡所有因素。下節說明如何評估 EXPLAIN 計劃來識別 ALL 分佈的候選者。
5. 對其餘資料表使用 AUTO 分佈。

   如果資料表大部分去正規化且未參與聯結，或如果您未明確選擇另一個分佈樣式，請使用 AUTO 分佈。

若要讓 Amazon Redshift 選擇適當的分佈樣式，請不要明確指定分佈樣式。

評估查詢計畫

您可以使用查詢計劃來識別可將分佈樣式最佳化的候選者。

在做出初始設計決策之後，請建立資料表、將資料載入其中，然後測試它們。使用儘可能接近真實資料的測試資料集。測量載入時間做為比較的基準。

評估哪些查詢代表您預期執行成本最高的查詢，尤指使用聯結和彙總的查詢。比較各種設計選項的執行期。比較執行期時，請不要將第一次執行查詢的時間列入計算，因為第一次執行期包含編譯時間。

DS_DIST_NONE

不需要重新分佈，因為對應配量是在運算節點上共置。通常您將只有一個 DS_DIST_NONE 步驟，即事實資料表與某個維度資料表之間的聯結。

DS_DIST_ALL_NONE

不需要重新分佈，因為內部資料表已使用 DISTSTYLE ALL。整個資料表位於每個節點上。

DS_DIST_INNER

重新配送內部資料表。

DS_DIST_OUTER

重新配送外部資料表。

DS_BCAST_INNER

將整個內部資料表的副本播送至所有運算節點。

DS_DIST_ALL_INNER

因為外部資料表使用 DISTSTYLE ALL，會將整個內部資料表重新配送至單一配量。

DS_DIST_BOTH

重新配送這兩個資料表。

DS_DIST_NONE 和 DS_DIST_ALL_NONE 都是不錯的選擇。它們指出該步驟不需要重新分佈，因為所有聯結已共置。

DS_DIST_INNER 表示步驟將可能有相當高的成本，因為內部資料表正在重新分佈至節點。DS_DIST_INNER 指出外部資料表已適當地分佈在聯結索引鍵上。將內部資料表的分佈索引鍵設為聯結索引鍵，以將此項轉換為 DS_DIST_NONE。在某些情況下，無法在聯結索引鍵上分佈內部資料表，因為外部資料表不會在聯結索引鍵上分佈。如果是這種情況，請評估是否對內部資料表使用 ALL 分佈。如果資料表未經常或廣泛更新，而且大到足以帶來高重新分佈成本，請將分佈樣式變更為 ALL，然後重新測試。ALL 分佈會導致載入時間增加，因此當您重新測試時，請將載入時間併入您的評估因素中。

DS_DIST_ALL_INNER 不是好選擇。它表示整個內部資料表已重新分佈至單一配量，因為外部資料表使用 DISTSTYLE ALL，以便整個外部資料表的副本位於每個節點上。這會在單一節點上造成一連串無

效的聯結執行期，而不是使用所有節點來善用平行執行期。DISTSTYLE ALL 主要只用於內部聯結資料表。反之，指定分佈索引鍵或甚至對外部資料表使用分佈。

DS_BCAST_INNER 和 DS_DIST_BOTH 不是好選擇。通常，這些重新分佈之所以發生，原因是未在資料表的分佈索引鍵上聯結資料表。如果事實資料表尚未有分佈索引鍵，請將聯結資料欄指定為這兩個資料表的分佈索引鍵。如果事實資料表已在另一個資料欄上具有分佈索引鍵，則請評估變更分佈索引鍵來共置此聯結是否將改善整體效能。如果變更外部資料表的分佈索引鍵不是最佳選擇，則您可以對內部資料表指定 DISTSTYLE ALL 來實現共置。

下列範例顯示具有 DS_BCAST_INNER 和 DS_DIST_NONE 標籤之查詢計劃的一部分。

```
->  XN Hash Join DS_BCAST_INNER  (cost=112.50..3272334142.59 rows=170771 width=84)
        Hash Cond: ("outer".venueid = "inner".venueid)
        ->  XN Hash Join DS_BCAST_INNER  (cost=109.98..3167290276.71 rows=172456
 width=47)
              Hash Cond: ("outer".eventid = "inner".eventid)
              ->  XN Merge Join DS_DIST_NONE  (cost=0.00..6286.47 rows=172456 width=30)
                    Merge Cond: ("outer".listid = "inner".listid)
                    ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497
 width=14)
                    ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=24)
```

在變更維度資料表來使用 DISTSTYLE ALL 之後，相同查詢的查詢計劃便會顯示 DS_DIST_ALL_NONE 代替 DS_BCAST_INNER。另外，聯結步驟的相對成本會發生大幅變更。與之前查詢中的 3272334142.59 相比，總成本是 14142.59。

```
->  XN Hash Join DS_DIST_ALL_NONE  (cost=112.50..14142.59 rows=170771 width=84)
        Hash Cond: ("outer".venueid = "inner".venueid)
        ->  XN Hash Join DS_DIST_ALL_NONE  (cost=109.98..10276.71 rows=172456 width=47)
              Hash Cond: ("outer".eventid = "inner".eventid)
              ->  XN Merge Join DS_DIST_NONE  (cost=0.00..6286.47 rows=172456 width=30)
                    Merge Cond: ("outer".listid = "inner".listid)
                    ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497
 width=14)
                    ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=24)
```

查詢計劃範例

此範例顯示如何評估查詢計劃來找出最佳化分佈的機會。

搭配 EXPLAIN 命令執行下列查詢來產生查詢計劃。

```
explain
select lastname, catname, venuename, venuecity, venuestate, eventname,
month, sum(pricepaid) as buyercost, max(totalprice) as maxtotalprice
from category join event on category.catid = event.catid
join venue on venue.venueid = event.venueid
join sales on sales.eventid = event.eventid
join listing on sales.listid = listing.listid
join date on sales.dateid = date.dateid
join users on users.userid = sales.buyerid
group by lastname, catname, venuename, venuecity, venuestate, eventname, month
having sum(pricepaid)>9999
order by catname, buyercost desc;
```

在 TICKIT 資料庫中，SALES 是事實資料表，而 LISTING 是最大維度。為了共置資料表，SALES 會分佈在 LISTID (即 LISTING 的外部索引鍵) 上，而 LISTING 會分佈在其主索引鍵 LISTID 上。下列範例顯示 SALES 和 LISTING 的 CREATE TABLE 命令。

```
create table sales(
 salesid integer not null,
 listid integer not null distkey,
 sellerid integer not null,
 buyerid integer not null,
 eventid integer not null encode mostly16,
 dateid smallint not null,
 qtysold smallint not null encode mostly8,
 pricepaid decimal(8,2) encode delta32k,
 commission decimal(8,2) encode delta32k,
 saletime timestamp,
 primary key(salesid),
 foreign key(listid) references listing(listid),
 foreign key(sellerid) references users(userid),
 foreign key(buyerid) references users(userid),
 foreign key(dateid) references date(dateid))
        sortkey(listid,sellerid);

create table listing(
 listid integer not null distkey sortkey,
 sellerid integer not null,
 eventid integer not null encode mostly16,
 dateid smallint not null,
 numtickets smallint not null encode mostly8,
 priceperticket decimal(8,2) encode bytedict,
```

```
 totalprice decimal(8,2) encode mostly32,
 listtime timestamp,
 primary key(listid),
 foreign key(sellerid) references users(userid),
 foreign key(eventid) references event(eventid),
 foreign key(dateid) references date(dateid));
```

在下列查詢計劃中，SALES 和 LISTING 上聯結的合併聯結步驟顯示 DS_DIST_NONE，這指出步驟不需要重新分佈。不過，往上移動查詢計劃時，其他內部資料表顯示 DS_BCAST_INNER，這指出內部資料表將播送，做為查詢執行的一部分。因為只有一對資料表可以使用索引鍵分佈來進行共置，所以五個資料表必須重新播送。

```
QUERY PLAN
XN Merge  (cost=1015345167117.54..1015345167544.46 rows=1000 width=103)
  Merge Key: category.catname, sum(sales.pricepaid)
  ->  XN Network  (cost=1015345167117.54..1015345167544.46 rows=170771 width=103)
        Send to leader
        ->  XN Sort  (cost=1015345167117.54..1015345167544.46 rows=170771 width=103)
              Sort Key: category.catname, sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=15345150568.37..15345152276.08 rows=170771
 width=103)
                    Filter: (sum(pricepaid) > 9999.00)
                     ->  XN Hash Join DS_BCAST_INNER  (cost=742.08..15345146299.10
 rows=170771 width=103)
                           Hash Cond: ("outer".catid = "inner".catid)
                           ->  XN Hash Join DS_BCAST_INNER
 (cost=741.94..15342942456.61 rows=170771 width=97)
                                 Hash Cond: ("outer".dateid = "inner".dateid)
                                 ->  XN Hash Join DS_BCAST_INNER
 (cost=737.38..15269938609.81 rows=170766 width=90)
                                       Hash Cond: ("outer".buyerid = "inner".userid)
                                       ->  XN Hash Join DS_BCAST_INNER
 (cost=112.50..3272334142.59 rows=170771 width=84)
                                             Hash Cond: ("outer".venueid =
 "inner".venueid)
                                             ->  XN Hash Join DS_BCAST_INNER
 (cost=109.98..3167290276.71 rows=172456 width=47)
                                                   Hash Cond: ("outer".eventid =
 "inner".eventid)
                                                   ->  XN Merge Join DS_DIST_NONE
 (cost=0.00..6286.47 rows=172456 width=30)
                                                         Merge Cond: ("outer".listid =
 "inner".listid)
```

```
                                                        ->  XN Seq Scan on listing
(cost=0.00..1924.97 rows=192497 width=14)
                                                        ->  XN Seq Scan on sales
(cost=0.00..1724.56 rows=172456 width=24)
                                                  ->  XN Hash  (cost=87.98..87.98
rows=8798 width=25)
                                                        ->  XN Seq Scan on event
(cost=0.00..87.98 rows=8798 width=25)
                                              ->  XN Hash  (cost=2.02..2.02 rows=202
width=41)
                                                    ->  XN Seq Scan on venue
(cost=0.00..2.02 rows=202 width=41)
                                        ->  XN Hash  (cost=499.90..499.90 rows=49990
width=14)
                                              ->  XN Seq Scan on users
(cost=0.00..499.90 rows=49990 width=14)
                                  ->  XN Hash  (cost=3.65..3.65 rows=365 width=11)
                                        ->  XN Seq Scan on date  (cost=0.00..3.65
rows=365 width=11)
                            ->  XN Hash  (cost=0.11..0.11 rows=11 width=10)
                                  ->  XN Seq Scan on category  (cost=0.00..0.11 rows=11
width=10)
```

一種解決方案是將資料表修改為具有 DISTSTYLE ALL。

```
ALTER TABLE users ALTER DISTSTYLE ALL;
ALTER TABLE venue ALTER DISTSTYLE ALL;
ALTER TABLE category ALTER DISTSTYLE ALL;
ALTER TABLE date ALTER DISTSTYLE ALL;
ALTER TABLE event ALTER DISTSTYLE ALL;
```

再次搭配 EXPLAIN 執行相同的查詢，並檢查新的查詢計劃。聯結現在會顯示 DS_DIST_ALL_NONE，指出不需要重新分佈，因為已使用 DISTSTYLE ALL 將資料分佈至每一個節點。

```
QUERY PLAN
XN Merge  (cost=1000000047117.54..1000000047544.46 rows=1000 width=103)
  Merge Key: category.catname, sum(sales.pricepaid)
  ->  XN Network  (cost=1000000047117.54..1000000047544.46 rows=170771 width=103)
        Send to leader
        ->  XN Sort  (cost=1000000047117.54..1000000047544.46 rows=170771 width=103)
              Sort Key: category.catname, sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=30568.37..32276.08 rows=170771 width=103)
                    Filter: (sum(pricepaid) > 9999.00)
```

```
                    ->  XN Hash Join DS_DIST_ALL_NONE  (cost=742.08..26299.10
rows=170771 width=103)
                          Hash Cond: ("outer".buyerid = "inner".userid)
                          ->  XN Hash Join DS_DIST_ALL_NONE  (cost=117.20..21831.99
rows=170766 width=97)
                                Hash Cond: ("outer".dateid = "inner".dateid)
                                ->  XN Hash Join DS_DIST_ALL_NONE
(cost=112.64..17985.08 rows=170771 width=90)
                                      Hash Cond: ("outer".catid = "inner".catid)
                                      ->  XN Hash Join DS_DIST_ALL_NONE
(cost=112.50..14142.59 rows=170771 width=84)
                                            Hash Cond: ("outer".venueid =
"inner".venueid)
                                            ->  XN Hash Join DS_DIST_ALL_NONE
(cost=109.98..10276.71 rows=172456 width=47)
                                                  Hash Cond: ("outer".eventid =
"inner".eventid)
                                                  ->  XN Merge Join DS_DIST_NONE
(cost=0.00..6286.47 rows=172456 width=30)
                                                        Merge Cond: ("outer".listid =
"inner".listid)
                                                        ->  XN Seq Scan on listing
(cost=0.00..1924.97 rows=192497 width=14)
                                                        ->  XN Seq Scan on sales
(cost=0.00..1724.56 rows=172456 width=24)
                                                  ->  XN Hash  (cost=87.98..87.98
rows=8798 width=25)
                                                        ->  XN Seq Scan on event
(cost=0.00..87.98 rows=8798 width=25)
                                            ->  XN Hash  (cost=2.02..2.02 rows=202
width=41)
                                                  ->  XN Seq Scan on venue
(cost=0.00..2.02 rows=202 width=41)
                                      ->  XN Hash  (cost=0.11..0.11 rows=11 width=10)
                                            ->  XN Seq Scan on category
(cost=0.00..0.11 rows=11 width=10)
                                ->  XN Hash  (cost=3.65..3.65 rows=365 width=11)
                                      ->  XN Seq Scan on date  (cost=0.00..3.65
rows=365 width=11)
                          ->  XN Hash  (cost=499.90..499.90 rows=49990 width=14)
                                ->  XN Seq Scan on users  (cost=0.00..499.90 rows=49990
width=14)
```

分佈範例

下列範例顯示如何根據您在 CREATE TABLE 陳述式中定義的選項來分佈資料。

DISTKEY 範例

查看 TICKIT 資料庫中 USERS 資料表的結構描述。USERID 會定義為 SORTKEY 資料欄和 DISTKEY 資料欄：

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'users';

    column     |          type          | encoding | distkey | sortkey
---------------+------------------------+----------+---------+---------
 userid        | integer                | none     | t       |       1
 username      | character(8)           | none     | f       |       0
 firstname     | character varying(30)  | text32k  | f       |       0

...
```

USERID 是此資料表上分佈資料欄不錯的選擇。如果查詢 SVV_DISKUSAGE 系統檢視，您可以看到資料表很均勻地分佈。資料欄號碼以零開始，所以 USERID 為資料欄 0。

```
select slice, col, num_values as rows, minvalue, maxvalue
from svv_diskusage
where name='users' and col=0 and rows>0
order by slice, col;

slice| col | rows  | minvalue | maxvalue
-----+-----+-------+----------+----------
0    | 0   | 12496 | 4        | 49987
1    | 0   | 12498 | 1        | 49988
2    | 0   | 12497 | 2        | 49989
3    | 0   | 12499 | 3        | 49990
(4 rows)
```

資料表包含 49,990 個資料列。資料列 (num_values) 資料欄顯示每個配量包含大約相同數目的資料列。minvalue 和 maxvalue 資料欄顯示每一個配量上的值範圍。每一個配量幾乎包含整個值範圍，所以每一個配量都有很好的機會參與執行一個篩選使用者 ID 範圍的查詢。

此範例示範小型測試系統上的分佈。配量總數通常高得多。

如果您最常使用 STATE 資料欄進行聯結或分組，則可以選擇在 STATE 資料欄上進行分佈。下列範例顯示，若您建立一個新資料表，其中資料與 USERS 資料表相同，但將 DISTKEY 設為 STATE 資料欄。在此情況下，分佈不是平均的。配量 0 (13,587 個資料列) 比配量 3 (10,150 個資料列) 多保留大約 30% 的資料列。在規模很大的資料表中，此分佈扭曲數量可能對查詢處理具有負面影響。

```
create table userskey distkey(state) as select * from users;

select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'userskey' and col=0 and rows>0
order by slice, col;

slice | col | rows  | minvalue | maxvalue
------+-----+-------+----------+----------
    0 |   0 | 13587 |        5 |    49989
    1 |   0 | 11245 |        2 |    49990
    2 |   0 | 15008 |        1 |    49976
    3 |   0 | 10150 |        4 |    49986
(4 rows)
```

DISTSTYLE EVEN 範例

如果您建立一個新資料表，其中資料與 USERS 資料表相同，但將 DISTSTYLE 設為 EVEN，則資料列一律平均地分佈在配量上。

```
create table userseven diststyle even as
select * from users;

select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'userseven' and col=0 and rows>0
order by slice, col;

slice | col | rows  | minvalue | maxvalue
------+-----+-------+----------+----------
    0 |   0 | 12497 |        4 |    49990
    1 |   0 | 12498 |        8 |    49984
    2 |   0 | 12498 |        2 |    49988
    3 |   0 | 12497 |        1 |    49989
(4 rows)
```

不過，因為分佈不是根據特定資料欄，所以查詢處理能力可能降低，尤其在資料表聯結至其他資料表時更是如此。聯結資料欄若少了分佈，通常會影響可以有效地執行聯結操作的類型。當這兩個資料表在其各自聯結資料欄上進行分佈和排序時，會最佳化聯結、彙總和分組操作。

DISTSTYLE ALL 範例

如果您建立一個新資料表，其中資料與 USERS 資料表相同，但將 DISTSTYLE 設為 ALL，則所有資料列會分佈至每個節點的第一個配量。

```
select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'usersall' and col=0 and rows > 0
order by slice, col;

slice | col | rows  | minvalue | maxvalue
------+-----+-------+----------+----------
    0 |   0 | 49990 |        4 |    49990
    2 |   0 | 49990 |        2 |    49990

(4 rows)
```

使用排序索引鍵

> **Note**
>
> 建議您使用 SORTKEY AUTO 建立資料表。如果您這麼做，Amazon Redshift 會使用自動資料表最佳化來選擇排序索引鍵。如需詳細資訊，請參閱 使用自動資料表最佳化。本節的其餘部分會提供有關排序順序的詳細資訊。

或者，建立資料表時，您可以定義一或多個資料欄做為排序索引鍵。當資料最初載入至空的資料表時，資料列會依排序儲存在磁碟上。排序索引鍵資料欄的相關資訊會傳遞至查詢規劃器，而且規劃器會使用此資訊，來建構利用資料排序方式的計劃。如需詳細資訊，請參閱 CREATE TABLE。如需建立排序索引鍵時最佳作法的資訊，請參閱 選擇最佳的排序索引鍵。

排序可以有效率地處理限制範圍的述詞。Amazon Redshift 將單欄式資料儲存在 1 MB 磁碟區塊中。每個區塊的 min 和 max 值都會儲存為中繼資料的一部分。如果查詢使用範圍受限的述詞，則查詢處理器可以使用 min 和 max 值，在資料表掃描期間快速地略過大量區塊。例如，假設資料表儲存了依日期排序的五年資料，而查詢指定了一個月的日期範圍。在這種情況下，您最多可以從掃描中移除 98% 的磁碟區塊。如果未排序資料，則必須掃描更多的磁碟區塊 (可能全部)。

您可以指定複合或交錯排序索引鍵。當查詢述詞使用字首，即依序的排序索引鍵子集時，複合排序索引鍵更有效。交錯排序索引鍵對排序索引鍵中的每個資料欄都提供相等的權重，所以查詢述詞可以使用構成排序索引鍵之資料欄的任何子集，順序不拘。

若要了解選擇的排序索引鍵對查詢效能的影響，請使用 EXPLAIN 命令。如需詳細資訊，請參閱 查詢計劃和執行工作流程。

若要定義排序類型，請使用 INTERLEAVED 或 COMPOUND 關鍵字與 CREATE TABLE 或 CREATE TABLE AS 陳述式搭配。預設值為 COMPOUND。當您使用 INSERT、UPDATE 或 DELETE 操作定期更新資料表時，建議使用 COMPOUND。INTERLEAVED 排序索引鍵最多可以使用八個資料欄。視您的資料和叢集大小而定，VACUUM REINDEX 所需的時間會比 VACUUM FULL 大幅增加，因為它會額外進行交錯排序索引鍵分析。交錯資料表的排序和合併操作可能需要更長的時間，因為交錯排序可能需要重新排列比複合排序更多的資料列。

若要檢視資料表的排序索引鍵，請查詢 SVV_TABLE_INFO 系統檢視。

主題

多維資料配置排序 (預覽)

以下是預覽版中資料表的多維資料配置排序的發行前版本文件。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

> **Note**
>
> 此功能僅可透過預覽叢集或預覽工作群組使用。若要建立預覽叢集，請參閱 Amazon Redshift 管理指南中的建立預覽叢集。若要建立預覽工作群組，請參閱 Amazon Redshift 管理指南中的建立預覽工作群組。

多維資料配置排序索引鍵是一種 AUTO 排序索引鍵類型，以工作負載中找到的重複述詞為基礎。如果您的工作負載具有重複述詞，Amazon Redshift 可以透過共置滿足重複述詞的資料列來改善資料表掃描

效能。多維資料配置排序索引鍵會藉由分析工作負載中出現的重複述詞來存放資料，而不是以嚴格的資料欄順序存放資料表的資料。在工作負載中可以找到多個重複述詞。根據您的工作負載，這種排序索引鍵可以改善許多述詞的效能。Amazon Redshift 會自動判斷使用 AUTO 排序索引鍵定義的資料表是否應該使用排序索引鍵方法。

例如，假設您的資料表的資料是按照資料欄排序。可能需要檢查許多資料區塊，藉以判斷該資料是否符合工作負載中的述詞。但是，如果資料是按照述詞順序儲存在磁碟上，則需要掃描較少區塊以滿足查詢。在這種情況下，使用多維資料配置排序索引鍵是有好處的。

若要檢視查詢是否使用多維資料配置索引鍵，請參閱 SYS_QUERY_DETAIL 視觀表的 `step_attribute` 資料欄。當該值為 `multi-dimensional`，則將多維資料配置用於查詢。若要檢視以 AUTO 排序索引鍵定義的資料表是否使用多維資料配置，請參閱 `sortkey1` 視觀表的 SVV_TABLE_INFO 資料欄。當該值為 `padb_internal_mddl_key_col`，則將多維資料配置用於資料表排序索引鍵。

若要防止 Amazon Redshift 使用多維資料配置排序索引鍵，請選擇不同的資料表排序索引鍵選項。`SORTKEY AUTO`如需 SORTKEY 選項的詳細資訊，請參閱 CREATE TABLE。

複合排序索引鍵

複合索引鍵是由排序索引鍵定義中列出的所有資料欄組成，並依它們的列出順序排列。當查詢的篩選條件套用條件 (例如篩選條件和聯結) 來使用排序索引鍵的字首時，複合排序索引鍵最有用。當查詢僅依賴次要排序資料欄，而未參考主要資料欄時，複合排序的效能優勢會降低。COMPOUND 是預設排序類型。

複合排序索引鍵可加快聯結、GROUP BY 和 ORDER BY 操作，以及使用 PARTITION BY 和 ORDER BY 的視窗函數。例如，在聯結資料欄上分佈和預先排序資料時，通常比雜湊聯結還要快的合併聯結更為可行。複合排序索引鍵也可協助改善壓縮。

當您將資料列新增至已包含資料的已排序資料表時，未排序的區域會擴大，因而對效能產生重大影響。當資料表使用交錯排序時，效果更明顯，尤其是排序資料欄包含單調增加的資料 (例如日期或時間戳記資料欄) 時更明顯。執行 VACUUM 操作 (尤其在大型資料載入之後更應執行)，來重新排序和重新分析資料。如需詳細資訊，請參閱 管理未排序區域的大小。在清空以重新排序資料之後，建議做法是執行 ANALYZE 命令來更新查詢規劃器的統計中繼資料。如需詳細資訊，請參閱 分析資料表。

交錯排序索引鍵

交錯排序可對排序索引鍵中的每個資料欄或資料欄子集提供相等的權重。如果多個查詢使用不同資料欄進行篩選，則您通常可以使用交錯排序樣式來改善那些查詢的效能。當查詢在次要排序資料欄上使用限制性述詞時，相較於複合排序，交錯排序可大幅改善查詢效能。

> **Important**
>
> 不要在具有依序增加屬性 (如身分資料欄、日期或時間戳記) 的資料欄上使用交錯排序索引鍵。

透過實作交錯排序索引鍵所獲得的效能改善，應該與增加的載入和清空時間進行權衡。

交錯排序與高度選擇性查詢搭配最有效，因為這些查詢會在 WHERE 子句中的一個或多個排序索引鍵資料欄上進行篩選，例如 `select c_name from customer where c_region = 'ASIA'`。隨著受限之排序資料欄的數目增加，交錯排序的優勢更明顯。

使用大型資料表時，交錯排序更有效。排序會套用至每個配量。因此，當資料表足夠大到每個配量需要多個 1 MB 區塊時，交錯排序最有效。在這裡，查詢處理器可以使用限制性述詞跳過區塊的顯著比例。若要檢視資料表使用的區塊數目，請查詢 STV_BLOCKLIST 系統檢視。

在單一資料欄上進行排序時，如果資料欄值具有很長的常用字首，則交錯排序可能提供比複合排序更好的效能。例如，URL 通常以 "http://www" 開頭。複合排序索引鍵使用字首中數目受限的字元，因而導致許多索引鍵重複。交錯排序會對區域映射值使用內部壓縮方法，讓它們更能區分常用字首很長的資料欄值。

將 Amazon Redshift 佈建的叢集遷移至 Amazon Redshift Serverless 時，Redshift 會將具有交錯排序索引鍵和 DISTSTYLE KEY 的資料表轉換為複合排序索引鍵。DiSTSTYLE 則不會變更。如需分佈樣式的相關資訊，請參閱使用資料分佈樣式。

VACUUM REINDEX

當您將資料列新增至已包含資料的已排序資料表時，效能可能會在一段時間後惡化。複合排序和交錯排序都會發生這種惡化，但是對交錯資料表的影響更大。VACUUM 會還原排序，但對於交錯資料表，此操作可能需要更長的時間，因為合併新的交錯資料可能涉及修改每一個資料區塊。

最初載入資料表時，Amazon Redshift 會分析排序索引鍵資料欄中值的分佈，並使用該資訊以取得排序索引鍵資料欄的最佳交錯。當資料表越來越大時，排序索引鍵資料欄中值的分佈可能會變更或扭曲，尤其是日期或時間戳記資料欄。如果扭曲變得太大，效能可能會受到影響。若要重新分析排序索引鍵並還原效能，請搭配 REINDEX 關鍵字執行 VACUUM 命令。因為它對資料必須進行額外的分析階段，所以對於交錯資料表，VACUUM REINDEX 可能需要比標準 VACUUM 還要長的時間。若要檢視索引鍵分佈扭曲及上次重建索引時間的相關資訊，請查詢 SVV_INTERLEAVED_COLUMNS 系統檢視。

如需如何判斷執行 VACUUM 的頻率與執行 VACUUM REINDEX 的時間之相關資訊，請參閱決定是否重建索引。

定義資料表限制

唯一性、主索引鍵和外部索引鍵限制僅供參考，Amazon Redshift 不會在您填入資料表時強制執行它們。例如，如果您將資料插入具有相依性的資料表中，即使插入違反限制也可以成功執行。儘管如此，主索引鍵和外部索引鍵仍會做為規劃提示，而且如果您的 ETL 程序或應用程式中的某些其他程序強制其完整性，則應該宣告它們。

例如，查詢規劃工具會在特定統計計算中使用主索引鍵和外部索引鍵。這樣做是為了推斷影響子查詢裝飾關係技術的唯一性和參考關係。透過這樣做，它可以排序大量聯結並刪除多餘的聯結。

規劃器會運用這些索引鍵關係，但其假設 Amazon Redshift 資料表中的所有索引鍵與載入時一樣有效。如果您的應用程式允許無效的外部索引鍵或主索引鍵，則有些查詢可能傳回不正確的結果。例如，如果主索引鍵不是唯一的，則 SELECT DISTINCT 查詢可能傳回重複的資料列。如果您懷疑其有效性的話，請勿對您的資料表定義索引鍵限制。但是，當您知道主索引鍵和外部索引鍵有效時，您應該一律宣告它們和限制唯一性。

Amazon Redshift 的確會強制執行 NOT NULL 欄限制條件。

如需資料表限制條件的相關資訊，請參閱CREATE TABLE。如需如何捨棄具有相依性之資料表的資訊，請參閱 DROP TABLE。

載入資料

主題

COPY 命令是載入資料表的最有效方式。您也可以使用 INSERT 命令將資料新增至您的資料表，但與使用 COPY 相較，此方式效率明顯較低。COPY 命令能夠同時從多個資料檔案或多個資料串流中進行讀取。Amazon Redshift 會將工作負載配置到叢集節點並平行執行載入作業，包括排序資料列和在節點配量間配送資料。

> **Note**
>
> Amazon Redshift Spectrum 外部資料表處於唯讀狀態。您無法 COPY 或 INSERT 至外部資料表。

若要存取其他 AWS 資源上的資料，您的叢集必須具有存取這些資源的權限，以及執行必要動作才能存取資料。您可以使用 AWS Identity and Access Management (IAM) 限制使用者對叢集資源和資料的存取權限。

初始資料載入之後，如果您新增、修改或刪除大量資料，在刪除之後，您後續應該執行 VACUUM 命令來重新組織您的資料和回收空間。您也應該執行 ANALYZE 命令來更新資料表統計資料。

本節說明如何載入資料和對資料載入進行故障診斷，以及呈現用於載入資料的最佳實務。

使用 COPY 命令載入資料

主題

COPY 命令會利用 Amazon Redshift 大量平行處理 (MPP) 架構從 Amazon S3 上的檔案、從 DynamoDB 資料表，或從一或多個遠端主機的文字輸出平行讀取和載入資料。

> **Note**
>
> 我們強烈建議使用 COPY 命令來載入大量資料。使用個別 INSERT 陳述式填入資料表的速度可能會相當慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用 INSERT INTO ... SELECT 或 CREATE TABLE AS 來改善效能。如需詳細資訊，請參閱 INSERT 或 CREATE TABLE AS。

若要從其他 AWS 資源載入資料，您的叢集必須具有存取資源和執行必要動作的權限。

若要授予或撤銷使用 COPY 命令將資料載入至資料表的權限，請授予或撤銷 INSERT 權限。

您的資料必須採用適當的格式，才能載入至您的 Amazon Redshift 資料表。本節說明用於在載入您的資料之前準備和驗證資料，以及執行之前用於驗證 COPY 陳述式的準則。

若要保護您的檔案中的資訊，您可以先將資料檔案加密再將其上傳至您的 Amazon S3 儲存貯體；COPY 會在執行載入時解密資料。您也可以對使用者提供暫時的安全性登入資料，來限制對您的載入資料的存取。暫時安全性登入資料提供加強的安全性，因為有效期限較短，且過期之後不能重複使用。

Amazon Redshift 具有內建的 COPY 功能，可快速載入未壓縮的已分隔資料。但您可以使用 gzip、lzop 或 bzip2 壓縮檔案來節省上傳檔案的時間。

如果 COPY 查詢中有下列關鍵字，則不支援自動分割未壓縮的資料：ESCAPE、REMOVEQUOTES 和 FIXEDWIDTH。但是支援 CSV 關鍵字。

為了協助確保您在 AWS 雲端傳輸中的資料安全，Amazon Redshift 使用硬體加速 SSL 與 Amazon S3 或 Amazon DynamoDB 通訊，以進行複製、卸載、備份和還原作業。

直接從 Amazon DynamoDB 資料表載入您的資料表時，您可以選擇控制所使用的 Amazon DynamoDB 佈建輸送量。

您可以選擇性地讓 COPY 分析您的輸入資料，和隨著載入程序自動套用最佳壓縮編碼至您的資料表。

登入資料和存取許可

若要使用其他 AWS 資源 (例如 Amazon S3、Amazon DynamoDB、Amazon EMR 或 Amazon EC2) 載入或卸載資料，您的叢集必須具有存取資源和執行必要動作存取資料的權限。例如，若要從 Amazon S3 載入資料，COPY 必須具備儲存貯體的 LIST 存取權和儲存貯體物件的 GET 存取權。

叢集必須經過驗證，才能獲得授權來存取資源。您可以選擇角色型存取控制或金鑰型存取控制中的一個。本節說明這兩個方法的概觀。如需詳細資訊和範例，請參閱存取其他 AWS 資源的許可。

角色類型存取控制

使用角色型存取控制時，叢集會暫時代表您擔任 AWS Identity and Access Management (IAM) 角色。然後，根據授與角色的授權，您的叢集可以存取所需的 AWS 資源。

我們建議您使用以角色為基礎的存取控制，因為除了保護您的認證之外，還可以更安全、更精細地控制 AWS 資源和敏感使用者資料的存取。 AWS

若要使用角色型存取控制，您必須先使用 Amazon Redshift 服務角色類型建立 IAM 角色，再將該角色連接至叢集。角色至少必須具備 COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可中列出的許可。如需建立 IAM 角色並將其附加到叢集的步驟，請參閱《Amazon Redshift 管理指南》中的建立 IAM 角色以允許您的 Amazon Redshift 叢集存取 AWS 服務。

您可以使用 Amazon Redshift 管理主控台、CLI 或 API 將角色新增至叢集，或檢視與叢集相關聯的角色。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的使用 IAM 角色授權 COPY 和 UNLOAD 操作。

建立 IAM 角色時，IAM 會傳回角色的 Amazon Resource Name (ARN)。若要使用 IAM 角色來執行 COPY 命令，請使用 IAM_ROLE 參數或 CREDENTIALS 參數來提供角色 ARN。

下列 COPY 命令範例會使用 IAM_ROLE 參數搭配角色 `MyRedshiftRole` 進行身分驗證。

```
copy customer from 's3://mybucket/mydata'
iam_role 'arn:aws:iam::12345678901:role/MyRedshiftRole';
```

使 AWS 用者至少必須具有中列出的權限COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可。

金鑰型存取控制

透過以金鑰為基礎的存取控制，您可以為獲得授權存取包含資料之 AWS 資源的使用者提供存取金鑰 ID 和秘密存取金鑰。

> **Note**
>
> 強烈建議使用 IAM 角色來驗證身分，而不要提供純文字存取金鑰 ID 和私密存取金鑰。如果您選擇以金鑰為基礎的存取控制，請勿使用您的 AWS 帳號 (root) 憑證。請一律建立 IAM 使用者，並提供該使用者的存取金鑰 ID 和私密存取金鑰。關於建立 IAM 使用者的步驟，請參閱在 AWS 帳戶中建立 IAM 使用者。

準備您的輸入資料

如果您的輸入資料與將接收它的資料表資料欄不相容，COPY 命令將會失敗。

使用下列準則來幫助確保您的輸入資料有效：

- 您的資料只可以包含長度最多四個位元組的 UTF-8 字元。
- 驗證 CHAR 和 VARCHAR 字串的長度未多於對應資料欄的長度。VARCHAR 字串是以位元組而非字元測量，因此，以中文字元四字元字串而言，每個會佔用四個位元組，所以需要 VARCHAR(16) 資料欄。
- 多位元組字元只可以搭配 VARCHAR 資料欄使用。驗證多位元組字元長度不超過四個位元組。
- 驗證 CHAR 資料欄的資料僅包含單位元組字元。

- 請勿包含任何特殊字元或語法來指出記錄中的最後一個欄位。此欄位可以是分隔符號。
- 如果您的資料包含 null 結束字元，也稱為 NUL (UTF-8 0000) 或二進位零 (0x000)，您可以藉由在 COPY 命令中使用 NULL AS 選項，以 NULLS 的形式將這些字元載入至 CHAR 或 VARCHAR 資料欄：`null as '\0'` 或 `null as '\000'`。如果您不使用 NULL AS，null 結束字元將造成您的 COPY 失敗。
- 如果您的字串包含特殊字元，例如分隔符號和內嵌的換行字元，請使用 ESCAPE 選項搭配 COPY 命令。
- 驗證所有單引號和雙引號均正確成對。
- 驗證浮點字串採用標準浮點格式，例如 12.123，或指數格式，例如 1.0E4。
- 驗證所有時間戳記和日期字串遵循 DATEFORMAT 和 TIMEFORMAT 字串 的規格。預設的時間戳記格式為 YYYY-MM-DD hh:mm:ss，而預設的日期格式為 YYYY-MM-DD。
- 如需個別資料類型的界限和限制的相關資訊，請參閱資料類型。如需多位元組字元錯誤的詳細資訊，請參閱多位元組字元載入錯誤

從 Amazon S3 載入資料

主題

COPY 命令會利用 Amazon Redshift 大量平行處理 (MPP) 架構，從 Amazon S3 儲存貯體中的單一檔案或多個檔案平行讀取和載入資料。如果檔案經過壓縮，您可以將資料分割為多個檔案，以充分利用平行處理的優勢。(此規則有例外情況。如需詳細資訊，請參閱載入資料檔案。) 您也可以藉由在資料表上設定分佈索引鍵來善加利用平行處理。如需分佈索引鍵的相關資訊，請參閱使用資料分佈樣式。

資料會載入到目標資料表，每個資料列一行。資料檔案中的欄位會依次與資料表資料欄對應，由左至右。資料檔案中的欄位可以是固定寬度或以字元分隔；預設的分隔符號為管線 (|)。依預設，所有資料表資料欄已載入，但您可以選擇性地定義以逗點分隔的資料欄清單。如果資料表資料欄未包含在 COPY 命令指定的資料欄清單中，它會使用預設值載入。如需詳細資訊，請參閱 載入預設的欄位值。

從已壓縮和未壓縮的檔案載入資料

載入壓縮的資料時，建議您將每個資料表的資料分割為多個檔案。載入未壓縮的已分隔資料後，COPY 命令會使用大量平行處理 (MPP) 和掃描範圍，從 Amazon S3 儲存貯體中的大型檔案載入資料。

從多個壓縮的檔案載入資料

如果您有壓縮的資料，建議您將每個資料表的資料分割為多個檔案。COPY 命令可以從多個檔案平行載入資料。您可以透過為集合指定通用字首或字首索引鍵，或在資訊清單檔案中明確列出檔案來載入多個檔案。

將您的資料分割為檔案，使得檔案的數量為您的叢集中配量數量的倍數。以該方式，Amazon Redshift 可以在配量間平均分割資料。每一節點的配量數目取決於叢集的節點大小。例如，每個 dc2.large 運算節點都有兩個磁碟片段，而且每個 dc2.8xlarge 運算節點都有 16 個磁碟片段。如需每個節點大小有多少配量的相關資訊，請參閱《Amazon Redshift 管理指南》中的關於叢集和節點。

節點全部會參與在執行的平行查詢中，以處理盡可能平均地分佈於配量的資料。如果您有具有兩個 dc2.large 節點的叢集，您可能會將資料分割成四個檔案或四個的倍數。Amazon Redshift 在分割工作負載時，不會考量檔案大小。因此，您需要確定壓縮後的檔案大小大致相同 (從 1 MB 到 1 GB)。

若要使用物件字首來識別載入檔案，將每個檔案以通用字首命名。例如，`venue.txt` 檔案可以分割為四個檔案，如下所示。

```
venue.txt.1
venue.txt.2
venue.txt.3
venue.txt.4
```

如果您將多個檔案放置在儲存貯體的資料夾中，並將資料夾名稱指定為該字首，則 COPY 就會載入資料夾中的所有檔案。如果您使用資訊清單檔案明確列出要載入的檔案，檔案可以位在不同的儲存貯體或資料夾中。

如需資訊清單檔案的相關資訊，請參閱Example: COPY from Amazon S3 using a manifest。

從未壓縮的已分隔檔案載入資料

當您載入未壓縮的已分隔資料時，COPY 命令會使用 Amazon Redshift 中的大量平行處理 (MPP) 架構。Amazon Redshift 會自動使用平行運作的配量，以從 Amazon S3 儲存貯體中的大型檔案中載入資料範圍。檔案必須已分隔，才能進行平行載入。例如，以縱線字元分隔。使用 COPY 命令自動進行平行資料載入也可用於 CSV 檔案。您可以藉由在資料表上設定分佈索引鍵來利用平行處理。如需分佈索引鍵的相關資訊，請參閱使用資料分佈樣式。

若 COPY 查詢包含下列任何關鍵字，則不支援自動平行資料載入：ESCAPE、REMOVEQUOTES 和 FIXEDWIDTH。

來自檔案的資料會載入到目標資料表，每個資料列一行。資料檔案中的欄位會依次與資料表資料欄對應，由左至右。資料檔案中的欄位可以是固定寬度或以字元分隔；預設的分隔符號為管線 (|)。依預設，所有資料表資料欄已載入，但您可以選擇性地定義以逗點分隔的資料欄清單。如果資料表資料欄未包含在 COPY 命令指定的資料欄清單中，則會使用預設值載入。如需詳細資訊，請參閱 載入預設的欄位值。

如果資料未壓縮也未分隔時，請遵循以下一般程序從 Amazon S3 載入資料：

1. 將檔案上傳至 Amazon S3。
2. 執行 COPY 命令載入資料表。
3. 驗證資料已正確載入。

如需 COPY 命令的範例，請參閱 COPY 範例。如需載入 Amazon Redshift 的資料相關資訊，請檢查 STL_LOAD_COMMITS 和 STL_LOAD_ERRORS 系統資料表。

如需有關節點及每個節點所含配量的相關資訊，請參閱《Amazon Redshift 管理指南》中的關於叢集和節點。

將檔案上傳到 Amazon S3

主題

- 管理資料一致性
- 將加密資料上傳到 Amazon S3
- 驗證您的儲存貯體中存在正確的檔案

將文字檔案上傳到 Amazon S3 時，有幾個可採取的方法：

- 如果您有壓縮的檔案，建議您分割大型檔案以利用 Amazon Redshift 中的平行處理。
- 另一方面，COPY 會自動分割大型的未壓縮文字分隔檔案資料，以加速平行處理，並有效地從大型檔案分配資料。

建立 Amazon S3 儲存貯體來保存您的資料檔案，然後將資料檔案上傳至儲存貯體。如需有關建立儲存貯體及上傳檔案的詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的使用 Amazon S3 儲存貯體。

> **Important**
>
> 必須在與您叢集所在的相同 AWS 區域中建立存放資料檔案的 Amazon S3 儲存貯體，除非您使用 REGION 選項來指定 Amazon S3 儲存貯體所在的區域。

確定 S3 IP 範圍已新增至您的允許清單。若要進一步了解所需的 S3 IP 範圍，請參閱網路隔離。

透過在建立儲存貯體時使用 Amazon S3 主控台選取區域，或是在建立儲存貯體時使用 Amazon S3 API 或 CLI 來指定端點，即可以在特定區域中建立 Amazon S3 儲存貯體。

在資料載入之後，驗證 Amazon S3 上有正確的檔案。

管理資料一致性

Amazon S3 針對所 AWS 有區域的 Amazon S3 儲存貯體上的複製、卸載、插入 (外部表)、建立外部表格 AS 和 Amazon Redshift Spectrum 操作提供了強大的 read-after-write 一致性。此外，Amazon S3 Select、Amazon S3 存取控制清單、Amazon S3 物件標籤和物件中繼資料 (例如，HEAD 物件) 上的讀取操作均高度一致。如需資料一致性的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 Amazon S3 資料一致性模式。

將加密資料上傳到 Amazon S3

Amazon S3 同時支援伺服器端加密和用戶端加密。此主題討論伺服器端和用戶端加密之間的差異，並說明將用戶端加密與 Amazon Redshift 搭配使用的步驟。伺服器端加密對 Amazon Redshift 來說是透明的。

伺服器端加密

伺服器端加密是靜態資料加密，也就是說，Amazon S3 會在上傳資料時加密您的資料，並在您存取資料時將資料解密。使用 COPY 命令載入資料表時，與從 Amazon S3 上伺服器端加密或未加密的物件載入時沒有差異。如需伺服器端加密的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的使用伺服器端加密。

用戶端加密

在用戶端加密中，您的用戶端應用程式會管理您的資料的加密、加密金鑰和相關工具。您可以使用用戶端加密將資料上傳至 Amazon S3 儲存貯體，然後使用 COPY 命令載入資料搭配 ENCRYPTED 選項和私有加密金鑰，以提供更高的安全性。

您可以使用封套加密來加密您的資料。利用封套加密，您的應用程式會以獨佔方式處理所有加密。您的私密加密金鑰和未加密的資料永遠不會傳送到 AWS，因此安全地管理加密金鑰非常重要。如果您遺失加密金鑰，您將無法解密資料，也無法從中 AWS復原加密金鑰。封套加密會結合快速對稱加密的效能，同時保有非對稱金鑰所提供金鑰管理的更高安全性。Amazon S3 加密用戶端會產生對稱金鑰 (信封對稱金鑰) 來加密您的資料，然後該金鑰會由根金鑰加密，並與資料一 one-time-use 起存放在 Amazon S3 中。Amazon Redshift 在載入期間存取您的資料時，系統會使用您的實際金鑰來擷取並解密加密的對稱金鑰，然後將資料解密。

若要在 Amazon Redshift 中使用 Amazon S3 用戶端加密資料，請參閱《Amazon Simple Storage Service 使用者指南》的使用用戶端加密保護資料來遵循所述步驟，並搭配您使用的其他要求：

- 對稱加密 – 適用於 Java 的 AWS 開發套件 `AmazonS3EncryptionClient` 類別使用上述基於對稱金鑰加密的封套加密程序。使用此類別來建立 Amazon S3 用戶端，以上傳用戶端加密的資料。
- 256 位元 AES 根對稱金鑰 – 根金鑰會加密信封金鑰。您可以將根金鑰傳遞至您的 `AmazonS3EncryptionClient` 類別執行個體。儲存此金鑰，因為您將需要用它將資料複製到 Amazon Redshift。
- 用來儲存加密信封金鑰的物件中繼資料 – 依預設，Amazon S3 會將信封金鑰儲存為 `AmazonS3EncryptionClient` 類別的物件中繼資料。儲存為物件中繼資料的加密信封金鑰會在解密程序期間使用。

> **Note**
>
> 若您在初次使用加密 API 時得到密碼套件加密錯誤訊息，您的 JDK 版本可能包含將加密及解密轉換之金鑰長度上限限制為 128 位元的 Java Cryptography Extension (JCE) 管轄權政策檔案。如需解決此問題的相關資訊，請參閱 Amazon 簡單儲存服務使用者指南中的使用 AWS SDK for Java 指定用戶端加密。

如需使用 COPY 命令將用戶端加密檔案載入至您的 Amazon Redshift 資料表的詳細資訊，請參閱從 Amazon S3 載入加密的資料檔案。

範例：上傳用戶端加密資料

如需如何使用 Java AWS SDK 上傳用戶端加密資料的範例，請參閱 Amazon 簡單儲存服務使用者指南中的使用用戶端加密保護資料。

第二個選項會顯示在用戶端加密期間要讓資料可以在 Amazon Redshift 中載入，您必須進行的選擇。具體來說，此範例會顯示使用物件中繼資料來儲存加密的信封金鑰和使用 256 位元 AES 根對稱金鑰。

此範例提供範例程式碼，使用 AWS SDK to Java 建立 256 位元 AES 對稱根金鑰並將其儲存至檔案。然後該範例會使用可先在用戶端上先加密樣本資料的 S3 加密用戶端，將物件上傳至 Amazon S3。此範例也會下載物件，並驗證資料一致。

驗證您的儲存貯體中存在正確的檔案

將您的檔案上傳至您的 Amazon S3 儲存貯體之後，我們建議列出儲存貯體的內容，以驗證已包含所有正確的檔案，並且沒有不需要的檔案。例如，如果儲存貯體 `mybucket` 保存名為 `venue.txt.back` 的檔案，下列命令會載入該檔案 (可能非特意)：

```
copy venue from 's3://mybucket/venue' … ;
```

如果您想要具體控制載入的檔案，您可以使用資訊清單檔案來明確列出資料檔案。如需使用資訊清單檔案的相關資訊，請參閱 COPY 命令的 copy_from_s3_manifest_file 選項和 COPY 範例中的 Example: COPY from Amazon S3 using a manifest。

如需列出儲存貯體內容的相關資訊，請參閱《Amazon S3 開發人員指南》中的列出物件金鑰。

使用 COPY 命令從 Amazon S3 載入

主題

- 使用資訊清單指定資料檔案
- 從 Amazon S3 載入壓縮的資料檔案
- 從 Amazon S3 載入固定寬度資料
- 從 Amazon S3 載入多位元組資料
- 從 Amazon S3 載入加密的資料檔案

使用 COPY 命令從 Amazon S3 上的資料檔案平行載入資料表。您可以透過使用 Amazon S3 物件字首或使用資訊清單檔案來指定要載入的檔案。

使用字首指定要載入之檔案的語法如下所示：

```
copy <table_name> from 's3://<bucket_name>/<object_prefix>'
authorization;
```

資訊清單檔案為 JSON 格式檔案，列出要載入的資料檔案。使用資訊清單檔案指定要載入之檔案的語法如下所示：

```
copy <table_name> from 's3://<bucket_name>/<manifest_file>'
authorization
manifest;
```

要載入的資料表必須已存在於資料庫中。如需建立資料表的詳細資訊，請參閱 SQL 參考中的 CREATE TABLE。

授權值提供叢集存取 Amazon S3 物件所需的 AWS 授權。如需所需許可的詳細資訊，請參閱COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可。身分驗證的偏好方法是指定 IAM_ROLE 參數，並提供 IAM 角色的 Amazon Resource Name (ARN) 所需的許可。如需詳細資訊，請參閱角色類型存取控制。

若要使用 IAM_ROLE 參數進行驗證，請取代 *<aws-account-id>* 和 *<role-name>*，如下列語法所示。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

下列範例顯示使用 IAM 角色進行身分驗證。

```
copy customer
from 's3://mybucket/mydata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

如需其他授權選項的相關資訊，請參閱授權參數

如果想要驗證您的資料而不實際載入資料表，請使用 NOLOAD 選項搭配 COPY 命令。

下列範例顯示名為 venue.txt 的檔案中以管線分隔資料的前幾個資料列。

```
1|Toyota Park|Bridgeview|IL|0
2|Columbus Crew Stadium|Columbus|OH|0
3|RFK Stadium|Washington|DC|0
```

上傳檔案至 Amazon S3 之前，將檔案分割為多個檔案，使得 COPY 命令可以使用平行處理載入它。檔案數量應為您叢集中的分割的倍數。分割您的載入資料，使檔案皆有相同的大小，在壓縮之後介於 1 MB 至 1 GB。如需詳細資訊，請參閱 從已壓縮和未壓縮的檔案載入資料。

例如，venue.txt 檔案可以分割為四個檔案，如下所示：

```
venue.txt.1
```

```
venue.txt.2
venue.txt.3
venue.txt.4
```

下列 COPY 命令會使用資料檔案中以管線分隔的資料來載入 VENUE 資料表，具有 Amazon S3 儲存貯體 mybucket 中的字首 'venue'。

> **Note**
>
> 下列範例中的 Amazon S3 儲存貯體 mybucket 不存在。如需使用現有 Amazon S3 儲存貯體中實際資料的 COPY 命令範例，請參閱載入範例資料。

```
copy venue from 's3://mybucket/venue'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

如果不存在具有 'venue' 金鑰前綴的任何 Amazon S3 物件，則載入會失敗。

使用資訊清單指定資料檔案

您可以使用清單檔案來確保 COPY 命令會載入所有必要檔案 (且只有必要檔案) 進行資料載入。您可以使用資訊清單從不同儲存貯體載入檔案，或載入不共用相同字首的檔案。不要為 COPY 命令提供物件路徑，而是提供明確列出要載入之檔案的 JSON 格式文字檔案名稱。資訊清單檔案中的 URL 必須指定儲存貯體名稱以及檔案的完整物件路徑，而不只是字首。

如需資訊清單檔案的相關資訊，請參閱使用資訊清單來指定資料檔案 COPY 範例。

下列範例顯示的 JSON 會從不同的儲存貯體載入檔案，並具有開頭為日期戳記的檔案名稱。

```
{
  "entries": [
    {"url":"s3://mybucket-alpha/2013-10-04-custdata", "mandatory":true},
    {"url":"s3://mybucket-alpha/2013-10-05-custdata", "mandatory":true},
    {"url":"s3://mybucket-beta/2013-10-04-custdata", "mandatory":true},
    {"url":"s3://mybucket-beta/2013-10-05-custdata", "mandatory":true}
  ]
}
```

選用的 mandatory 旗標會指定找不到檔案時 COPY 是否應該傳回錯誤。mandatory 的預設值為 false。不考慮任何必要設定，只要找不到檔案，COPY 就會終止。

下列範例會執行 COPY 命令並搭配上一個範例中的資訊清單，名為 cust.manifest。

```
copy customer
from 's3://mybucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

使用 UNLOAD 建立的資訊清單

UNLOAD 操作使用 MANIFEST 參數建立的資訊清單可能會有 COPY 操作不需要的索引鍵。例如，以下 UNLOAD 清單檔案包含 meta 索引鍵，針對 Amazon Redshift Spectrum 外部資料表，還有載入 ORC 或 Parquet 檔案格式的資料檔案時，都需要此索引鍵。meta 索引鍵包含的 content_length 索引鍵具有的值為檔案的實際大小 (以位元組為單位)。COPY 操作僅需要 url 索引鍵和選用的 mandatory 索引鍵。

```
{
  "entries": [
    {"url":"s3://mybucket/unload/manifest_0000_part_00", "meta": { "content_length":
 5956875 }},
    {"url":"s3://mybucket/unload/unload/manifest_0001_part_00", "meta":
 { "content_length": 5997091 }}
 ]
}
```

如需資訊清單檔案的相關資訊，請參閱Example: COPY from Amazon S3 using a manifest。

從 Amazon S3 載入壓縮的資料檔案

若要載入使用 gzip、lzop 或 bzip2 壓縮的資料檔案，請併入對應的選項：GZIP、LZOP 或 BZIP2。

例如，下列命令會從使用 lzop 壓縮的檔案載入。

```
copy customer from 's3://mybucket/customer.lzo'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' lzop;
```

> **Note**
>
> 如果您使用 lzop 壓縮來壓縮資料檔案，並使用 -filter 選項，則 COPY 指令不支援它。

從 Amazon S3 載入固定寬度資料

固定寬度資料檔案其資料的每個資料欄長度一致。固定寬度資料檔案中的每個欄位有完全相同的長度和位置。針對固定寬度資料檔案中的字元資料 (CHAR 和 VARCHAR)，您必須包括前方或結尾空格做為預留位置以便維持寬度統一。針對整數，您必須使用前方零做為預留位置。固定寬度資料檔案沒有分隔符號可分隔資料欄。

若要將固定寬度資料檔案載入至現有資料表，請在 COPY 命令中使用 FIXEDWIDTH 參數。您的資料表規格必須符合 fixedwidth_spec 的值，資料才能正確載入。

若要從檔案載入固定寬度資料至資料表，請發出下列命令：

```
copy table_name from 's3://mybucket/prefix'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth 'fixedwidth_spec';
```

fixedwidth_spec 參數是字串，包含每個資料欄的識別碼以及每個資料欄的寬度，以冒號分隔。**column:width** 對組是使用逗號分隔。識別碼可以是您選擇的任何項目：數字、字母或兩者的結合。識別碼對資料表本身沒有關聯，所有規格必須以與資料表相同的順序包含資料欄。

下列兩個範例會顯示相同的規格，第一個使用數值識別碼，而第二個使用字串識別碼：

```
'0:3,1:25,2:12,3:2,4:6'
```

```
'venueid:3,venuename:25,venuecity:12,venuestate:2,venueseats:6'
```

下列範例顯示可以使用上述規格載入至 VENUE 資料表的固定寬度樣本資料：

```
1  Toyota Park               Bridgeview  IL0
2  Columbus Crew Stadium     Columbus    OH0
3  RFK Stadium               Washington  DC0
4  CommunityAmerica Ballpark Kansas City KS0
5  Gillette Stadium          Foxborough  MA68756
```

下列 COPY 命令會將此資料集載入至 VENUE 資料表：

```
copy venue
```

```
from 's3://mybucket/data/venue_fw.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth 'venueid:3,venuename:25,venuecity:12,venuestate:2,venueseats:6';
```

從 Amazon S3 載入多位元組資料

如果資料包含非 ASCII 多位元組字元 (例如中文或斯拉夫文字元)，您必須將資料載入 VARCHAR 欄。VARCHAR 資料類型支援四位元組 UTF-8 字元，但 CHAR 資料類型只接受單位元組 ASCII 字元。您無法將五位元組或更長的字元載入 Amazon Redshift 資料表。如需 CHAR 和 VARCHAR 的相關資訊，請參閱資料類型。

若要檢查輸入檔案使用的編碼，請使用 Linux *file* 命令：

```
$ file ordersdata.txt
ordersdata.txt: ASCII English text
$ file uni_ordersdata.dat
uni_ordersdata.dat: UTF-8 Unicode text
```

從 Amazon S3 載入加密的資料檔案

您可以使用 COPY 命令來載入使用伺服器端加密、用戶端加密 (或兩者) 上傳至 Amazon S3 的資料檔案。

COPY 命令支援下列類型的 Amazon S3 加密：

- 使用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密
- 伺服器端加密與 AWS KMS keys (SSE-KMS)
- 使用用戶端對稱根金鑰的用戶端加密

COPY 命令不支援下列類型的 Amazon S3 加密：

- 使用客戶提供金鑰 (SSE-C) 的伺服器端加密
- 用戶端加密 (使用) AWS KMS key
- 使用客戶提供非對稱根金鑰的用戶端加密

如需 Amazon S3 加密的相關資訊，請參閱《Amazon Simple Storage Service 開發人員指南》中的使用伺服器端加密保護資料和使用用戶端加密保護資料。

UNLOAD 命令會使用 SSE-S3 自動加密檔案。您也可以使用 SSE-KMS 或客戶管理對稱金鑰的用戶端加密來進行卸載。如需更多資訊，請參閱卸載加密的資料檔案

COPY 命令會自動識別和載入使用 SSE-S3 和 SSE-KMS 加密的檔案。您可以透過指定 ENCRYPTED 選項並提供金鑰值，以載入使用用戶端對稱根金鑰加密的檔案。如需詳細資訊，請參閱 將加密資料上傳到 Amazon S3。

若要載入用戶端加密的資料檔案，請使用 MASTER_SYMMETRIC_KEY 參數並包括 ENCRYPTED 選項來提供根金鑰值。

```
copy customer from 's3://mybucket/encrypted/customer'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted
delimiter '|';
```

若要載入使用 gzip、lzop 或 bzip2 壓縮的加密資料檔案，請併入 GZIP、LZOP 或 BZIP2 選項與根金鑰值和 ENCRYPTED 選項。

```
copy customer from 's3://mybucket/encrypted/customer'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted
delimiter '|'
gzip;
```

從 Amazon EMR 載入資料

您可以使用 COPY 命令從 Amazon EMR 叢集平行載入資料，而叢集設定為以固定寬度檔案、字元分隔檔案、CSV 檔案或 JSON 格式檔案的形式，將文字檔案寫入叢集的 Hadoop 分散式檔案系統 (HDFS)。

從 Amazon EMR 載入資料的程序

本節會逐步引導您完成從 Amazon EMR 叢集載入資料的程序。下列小節提供必須完成每個步驟的詳細資訊。

- 步驟 1：設定 IAM 許可

  建立 Amazon EMR 叢集和執行 Amazon Redshift COPY 命令的使用者必須具備必要的許可。

- 步驟 2：建立 Amazon EMR 叢集。

  設定叢集以輸出文字檔案至 Hadoop 分散式檔案系統 (HDFS)。您將需要 Amazon EMR 叢集 ID 和叢集的主要公有 DNS (主控叢集的 Amazon EC2 執行個體端點)。

- 步驟 3：擷取 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址

  公有金鑰可讓 Amazon Redshift 叢集節點與主機建立 SSH 連線。您將使用每個叢集節點的 IP 地址來設定主機安全群組，以允許使用這些 IP 地址從您的 Amazon Redshift 叢集存取。

- 步驟 4：將 Amazon Redshift 叢集公有金鑰新增至每個 Amazon EC2 主機的授權金鑰檔案

  您可以將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案，以便主機識別 Amazon Redshift 叢集並接受 SSH 連線。

- 步驟 5：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址

  修改 Amazon EMR 執行個體的安全群組，以新增傳入規則來接受 Amazon Redshift IP 地址。

- 步驟 6：執行 COPY 命令以載入資料

  從 Amazon Redshift 資料庫，執行 COPY 命令以將資料載入至 Amazon Redshift 資料表。

步驟 1：設定 IAM 許可

建立 Amazon EMR 叢集和執行 Amazon Redshift COPY 命令的使用者必須具備必要的許可。

設定 IAM 許可

1. 為將建立 Amazon EMR 叢集的使用者新增下列許可。

   ```
   ec2:DescribeSecurityGroups
   ec2:RevokeSecurityGroupIngress
   ec2:AuthorizeSecurityGroupIngress
   redshift:DescribeClusters
   ```

2. 為將執行 COPY 命令的 IAM 角色或使用者新增下列許可。

   ```
   elasticmapreduce:ListInstances
   ```

3. 將下列許可新增至 Amazon EMR 叢集的 IAM 角色。

   ```
   redshift:DescribeClusters
   ```

步驟 2：建立 Amazon EMR 叢集。

COPY 命令會從 Amazon EMR Hadoop 分散式檔案系統 (HDFS) 上的檔案載入資料。建立 Amazon EMR 叢集時，請設定叢集以輸出資料檔案至叢集的 HDFS。

建立 Amazon EMR 叢集

1. 在與 Amazon 紅移叢集相同的 AWS 區域中建立亞馬遜 EMR 叢集。

   如果 Amazon Redshift 叢集位於 VPC 中，Amazon EMR 叢集必須位在相同的 VPC 群組。如果 Amazon Redshift 叢集使用 EC2-Classic 模式 (亦即，不在 VPC 中)，Amazon EMR 叢集也必須使用 EC2-Classic 模式。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的在虛擬私有雲端 (VPC) 中管理叢集。
2. 設定叢集以將資料檔案輸出至叢集的 HDFS。HDFS 檔案名稱不能包括星號 (*) 或問號 (?)。

   > **Important**
   >
   > 檔案名稱不能包括星號 (*) 或問號 (?)。
3. 針對 Amazon EMR 叢集組態中的自動終止選項指定否，以便叢集在 COPY 命令執行時保持可用狀態。

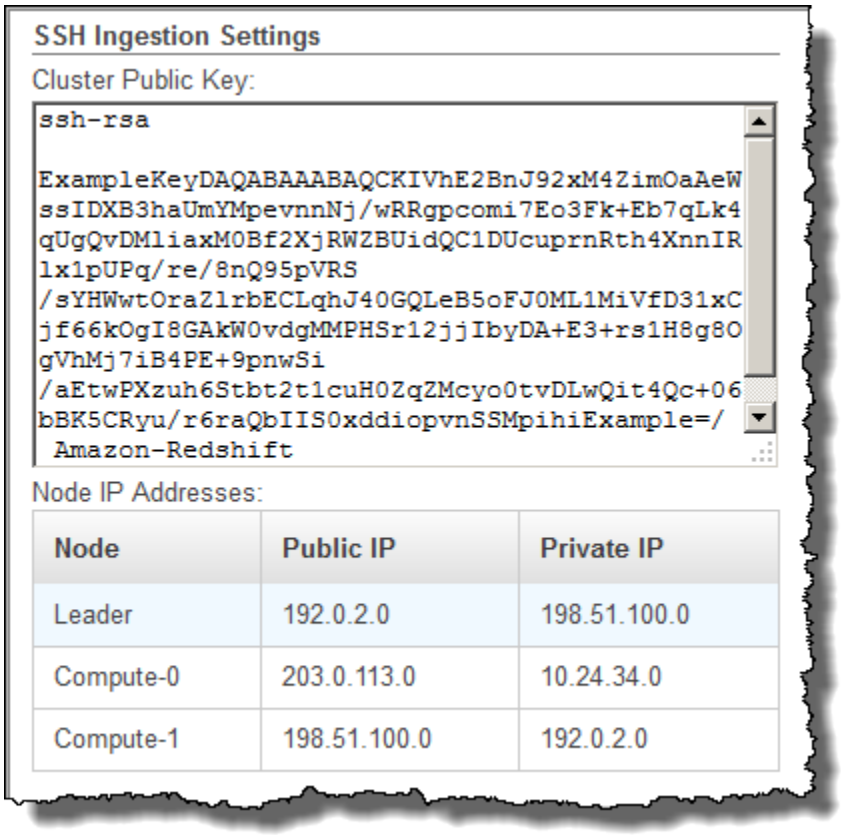
   > **Important**
   >
   > 如果 COPY 完成之前有任何資料檔案變更或刪除，可能會發生非預期的結果，COPY 操作也可能失敗。
4. 請記下叢集 ID 和主要公有 DNS (主控叢集的 Amazon EC2 執行個體端點)。您將在稍後的步驟中使用該資訊。

步驟 3：擷取 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址

使用主控台擷取您叢集的 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址

1. 存取 Amazon Redshift 管理主控台。
2. 在導覽窗格中，選擇叢集連結。
3. 從清單選取您的叢集。
4. 找到 SSH Ingestion Settings (SSH 擷取設定) 群組。

記下 Cluster Public Key (叢集公有金鑰) 和 Node IP addresses (節點 IP 地址)。您將在稍後的步驟中用到它們。



SSH Ingestion Settings

Cluster Public Key:

```
ssh-rsa

ExampleKeyDAQABAAABAQCKIVhE2BnJ92xM4ZimOaAeW
ssIDXB3haUmYMpevnnNj/wRRgpcomi7Eo3Fk+Eb7qLk4
qUgQvDMliaxM0Bf2XjRWZBUidQC1DUcuprnRth4XnnIR
lx1pUPq/re/8nQ95pVRS
/sYHWwtOraZlrbECLqhJ40GQLeB5oFJ0ML1MiVfD31xC
jf66kOgI8GAkW0vdgMMPHSr12jjIbyDA+E3+rs1H8g8O
gVhMj7iB4PE+9pnwSi
/aEtwPXzuh6Stbt2t1cuH0ZqZMcyo0tvDLwQit4Qc+06
bBK5CRyu/r6raQbIIS0xddiopvnSSMpihiExample=/
 Amazon-Redshift
```

Node IP Addresses:

Node	Public IP	Private IP
Leader	192.0.2.0	198.51.100.0
Compute-0	203.0.113.0	10.24.34.0
Compute-1	198.51.100.0	192.0.2.0

您將使用步驟 3 的私有 IP 地址來設定 Amazon EC2 主機以接受來自 Amazon Redshift 的連線。

若要使用 Amazon Redshift CLI 來擷取叢集的叢集公有金鑰和叢集節點 IP 地址，請執行 describe-clusters 命令。例如：

```
aws redshift describe-clusters --cluster-identifier <cluster-identifier>
```

回應將包含一個 ClusterPublicKey 值以及私有和公用 IP 位址清單，類似下列內容：

```
{
    "Clusters": [
        {
            "VpcSecurityGroups": [],
            "ClusterStatus": "available",
            "ClusterNodes": [
                {
```

```
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "LEADER",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-0",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-1",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                }
            ],
            "AutomatedSnapshotRetentionPeriod": 1,
            "PreferredMaintenanceWindow": "wed:05:30-wed:06:00",
            "AvailabilityZone": "us-east-1a",
            "NodeType": "dc2.large",
            "ClusterPublicKey": "ssh-rsa AAAABexamplepublickey...Y3TAl Amazon-
Redshift",
             ...
             ...
}
```

若要使用 Amazon Redshift API 來擷取叢集的叢集公有金鑰和叢集節點 IP 地址，請使用 `DescribeClusters` 動作。如需詳細資訊，請參閱 Amazon Redshift CLI 指南或亞馬 Amazon Redshift API 指南DescribeClusters中的描述叢集。

步驟 4：將 Amazon Redshift 叢集公有金鑰新增至每個 Amazon EC2 主機的授權金鑰檔案

您可以將叢集公有金鑰新增至所有 Amazon EMR 叢集節點的每個主機的授權金鑰檔案，以便主機識別 Amazon Redshift 並接受 SSH 連線。

將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案

1. 使用 SSH 連線存取主機。

   如需使用 SSH 連接至執行個體的詳細資訊，請參閱《Amazon EC2 使用者指南》中的連接至您的執行個體。
2. 從主控台或從 CLI 回應文字複製 Amazon Redshift 公有金鑰。

3. 將公有金鑰的內容複製和貼上至主機上的 /home/<ssh_username>/.ssh/authorized_keys 檔案。包括完整字串，包括字首 "ssh-rsa" 和字尾 "Amazon-Redshift"。例如：

```
ssh-rsa AAAACTP3isxgGzVWoIWpbVvRCOzYdVifMrh… uA70BnMHCaMiRdmvsDOedZDOedZ Amazon-
Redshift
```

步驟 5：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址

若要允許對主機執行個體的輸入流量，請編輯安全群組，並為每個 Amazon Redshift 叢集節點新增一個輸出規則。針對類別，對連接埠 22 上的 TCP 通訊協定選取 SSH。針對來源，輸入您在 步驟 3：擷取 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址 中擷取的 Amazon Redshift 叢集節點私有 IP 地址。如需將規則新增至 Amazon EC2 安全群組的相關資訊，請參閱《Amazon EC2 使用者指南》中的授權執行個體的傳入流量。

步驟 6：執行 COPY 命令以載入資料

執行 COPY 命令以連線至 Amazon EMR 叢集，並將資料載入至 Amazon Redshift 資料表。Amazon EMR 叢集必須繼續執行，直到 COPY 命令完成。例如，請勿將叢集設定為自動終止。

> **Important**
>
> 如果 COPY 完成之前有任何資料檔案變更或刪除，可能會發生非預期的結果，COPY 操作也可能失敗。

在 COPY 命令中，指定 Amazon EMR 叢集 ID 和 HDFS 檔案路徑和檔案名稱。

```
copy sales
from 'emr://myemrclusterid/myoutput/part*' credentials
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

您可以在檔案名稱引數中使用萬用字元星號 (*) 和問號 (?)。例如，part* 會載入檔案 part-0000、part-0001，以此類推。如果僅指定資料夾名稱，COPY 會嘗試載入該資料夾中的所有檔案。

> **Important**
>
> 如果使用萬用字元或只使用資料夾名稱，請確認不會載入不需要的檔案，否則 COPY 命令將會失敗。例如，某些程序可能將日誌檔案寫入至輸出資料夾。

從遠端主機載入資料

您可以使用 COPY 命令從一或多台遠端主機平行載入資料，例如 Amazon EC2 執行個體或其他電腦。COPY 會使用 SSH 連接至遠端主機，然後在遠端主機執行命令來產生文字輸出。

遠端主機可以是 Amazon EC2 Linux 執行個體，或另一台設定為接受 SSH 連線的 Unix 或 Linux 電腦。本指南假設您的遠端主機為 Amazon EC2 執行個體。當其他電腦上的程序不同時，指南將指出差異。

Amazon Redshift 可以連線到多台主機，而且可以對每台主機開啟多個 SSH 連線。Amazon Redshift 會透過每個連線傳送一個唯一的命令，將文字輸出產生到主機的標準輸出，然後 Amazon Redshift 會像讀取文字檔一樣讀取該輸出。

開始之前

開始之前，您應該備妥下列各項：

- 您可以使用 SSH 連線的一或多個主機，例如 Amazon EC2 執行個體。
- 主機上的資料來源。

  您將提供 Amazon Redshift 叢集將在主機上執行以產生文字輸出的命令。叢集連線至主機之後，COPY 命令會執行這些命令，讀取來自主機標準輸出的文字，以及並行載入資料至 Amazon Redshift 資料表。文字輸出必須採用 COPY 命令可以擷取的格式。如需更多資訊，請參閱準備您的輸入資料
- 從您的電腦存取主機。

  針對 Amazon EC2 執行個體，您將使用 SSH 連線來存取主機。您將需要存取主機以將 Amazon Redshift 叢集的公有金鑰新增至主機的授權金鑰檔案。
- 執行中的 Amazon Redshift 叢集。

  如需如何啟動叢集的相關資訊，請參閱《Amazon Redshift 入門指南》https://docs.aws.amazon.com/redshift/latest/gsg/。

載入資料程序

本節會逐步演練如何從從遠端主機載入資料的程序。下列小節提供每個步驟中必須完成的詳細資訊。

- 步驟 1：擷取叢集公有金鑰和叢集節點 IP 地址

  公有金鑰可讓 Amazon Redshift 叢集節點與遠端主機建立 SSH 連線。您將使用每個叢集節點的 IP 地址來設定主機安全群組或防火牆，以允許使用這些 IP 地址從您的 Amazon Redshift 叢集存取。

- 步驟 2:將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案

  您可以將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案，以便主機識別 Amazon Redshift 叢集並接受 SSH 連線。

- 步驟 3：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址

  針對 Amazon EC2，請修改執行個體的安全群組，以新增傳入規則來接受 Amazon Redshift IP 地址。針對其他主機，修改防火牆使得您的 Amazon Redshift 節點可以與遠端主機建立 SSH 連線。

- 步驟 4：取得主機的公有金鑰

  您可以選擇性地指定 Amazon Redshift 應該使用公有金鑰來識別主機。您必須找到公有金鑰，並將文字複製到您的資訊清單檔案。

- 步驟 5：建立資訊清單檔案

  清單檔案是 JSON 格式的文字檔案，具有 Amazon Redshift 連線至主機和擷取資料所需的詳細資訊。

- 步驟 6：上傳資訊清單檔案至 Amazon S3 儲存貯體

  Amazon Redshift 會讀取清單檔案，並使用該資訊來連線至遠端主機。如果 Amazon S3 儲存貯體不是位於與 Amazon Redshift 叢集所在的同一區域，您必須使用 REGION 選項來指定資料所在的區域。

- 步驟 7：執行 COPY 命令以載入資料

  從 Amazon Redshift 資料庫，執行 COPY 命令以將資料載入至 Amazon Redshift 資料表。

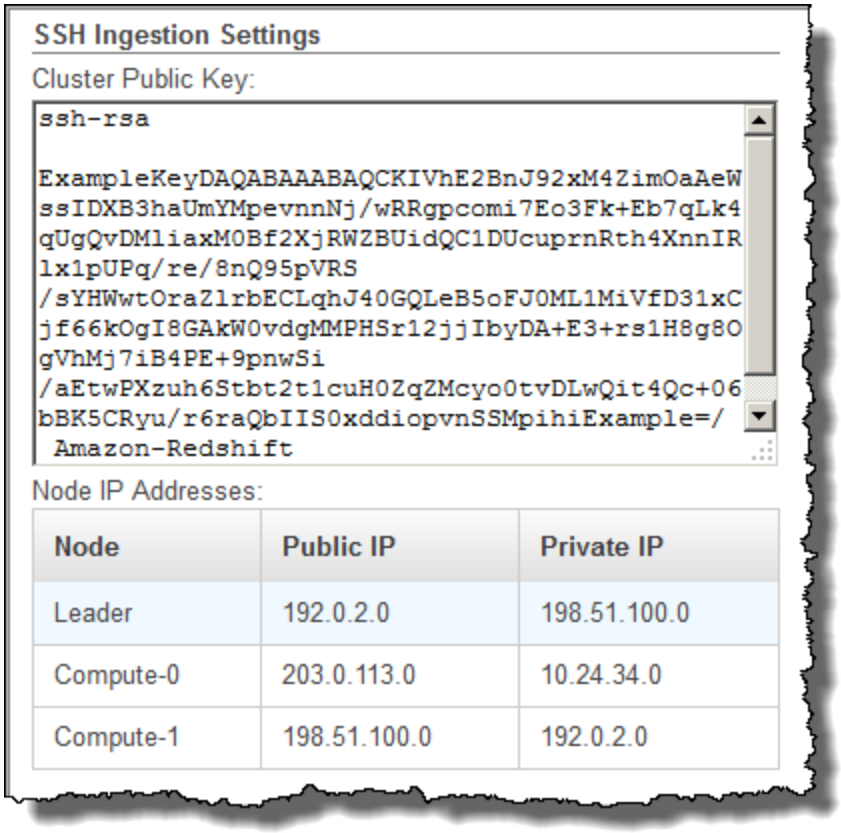
步驟 1：擷取叢集公有金鑰和叢集節點 IP 地址

使用主控台擷取您的叢集的叢集公有金鑰和叢集節點 IP 地址

1. 存取 Amazon Redshift 管理主控台。

2. 在導覽窗格中，選擇叢集連結。
3. 從清單選取您的叢集。
4. 找到 SSH Ingestion Settings (SSH 擷取設定) 群組。

   記下 Cluster Public Key (叢集公有金鑰) 和 Node IP addresses (節點 IP 地址)。您將在稍後的步驟中用到它們。



   您將使用步驟 3 的 IP 地址來設定主機以接受來自 Amazon Redshift 的連線。取決於您連線的主機類型以及它是否位於 VPC，您將使用公有 IP 地址或私有 IP 地址。

若要使用 Amazon Redshift CLI 來擷取叢集的叢集公有金鑰和叢集節點 IP 地址，請執行 describe-clusters 命令。

例如：

```
aws redshift describe-clusters --cluster-identifier <cluster-identifier>
```

回應將包括 ClusterPublicKey 和私人和公用 IP 位址清單，類似下列內容：

```
{
    "Clusters": [
        {
            "VpcSecurityGroups": [],
            "ClusterStatus": "available",
            "ClusterNodes": [
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "LEADER",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-0",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                },
                {
                    "PrivateIPAddress": "10.nnn.nnn.nnn",
                    "NodeRole": "COMPUTE-1",
                    "PublicIPAddress": "10.nnn.nnn.nnn"
                }
            ],
            "AutomatedSnapshotRetentionPeriod": 1,
            "PreferredMaintenanceWindow": "wed:05:30-wed:06:00",
            "AvailabilityZone": "us-east-1a",
            "NodeType": "dc2.large",
            "ClusterPublicKey": "ssh-rsa AAAABexamplepublickey...Y3TAl Amazon-
Redshift",
             ...
             ...
}
```

若要使用 Amazon Redshift API 擷取叢集的叢集公用金鑰和叢集節點 IP 地址，請使用以下 DescribeClusters 動作。如需詳細資訊，請參閱 Amazon Redshift CLI 指南或亞馬 Amazon Redshift API 指南DescribeClusters中的描述叢集。

步驟 2:將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案

您可以將叢集公有金鑰新增至每個主機的授權金鑰檔案，以便主機識別 Amazon Redshift 並接受 SSH 連線。

將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案

1. 使用 SSH 連線存取主機。

   如需使用 SSH 連接至執行個體的詳細資訊，請參閱《Amazon EC2 使用者指南》中的連接至您的執行個體。

2. 從主控台或從 CLI 回應文字複製 Amazon Redshift 公有金鑰。

3. 將公有金鑰的內容複製和貼上至遠端主機上的 /home/<ssh_username>/.ssh/authorized_keys 檔案。<ssh_username> 必須符合資訊清單檔案中 "username" 欄位的值。包括完整字串，包括字首 "ssh-rsa" 和字尾 "Amazon-Redshift"。例如：

```
ssh-rsa AAAACTP3isxgGzVWoIWpbVvRCOzYdVifMrh… uA70BnMHCaMiRdmvsDOedZDOedZ Amazon-
Redshift
```

步驟 3：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址

如果是使用 Amazon EC2 執行個體或 Amazon EMR 叢集，新增輸出規則至主機的安全群組可允許來自每個 Amazon Redshift 叢集節點的流量。針對類別，對連接埠 22 上的 TCP 通訊協定選取 SSH。針對來源，輸入您在 步驟 1：擷取叢集公有金鑰和叢集節點 IP 地址 中擷取的 Amazon Redshift 叢集節點 IP 地址。如需將規則新增至 Amazon EC2 安全群組的相關資訊，請參閱《Amazon EC2 使用者指南》中的授權執行個體的傳入流量。

使用私有 IP 地址的時機：

- 您有一個不在 Virtual Private Cloud (VPC) (VPC) 中的 Amazon Redshift 叢集，以及一個 Amazon EC2-經典執行個體，這兩個執行個體都位於同一個區域。 AWS
- 您有一個位於 VPC 中的 Amazon Redshift 叢集，以及一個 Amazon EC2-VPC 執行個體，這兩個執行個體都位於同一個 AWS 區域且位於相同的 VPC 中。

否則，請使用公有 IP 地址。

如需在 VPC 中使用 Amazon Redshift 的相關資訊，請參閱《Amazon Redshift 管理指南》中的在虛擬私有雲端 (VPC) 中管理叢集。

步驟 4：取得主機的公有金鑰

您可以選擇性地在資訊清單檔案中提供主機的公有金鑰，使得 Amazon Redshift 可以識別主機。COPY 命令不需要主機公有金鑰，但因為安全原因，我們強烈建議使用公有金鑰來幫助避免 'man-in--middle' 攻擊。

您可以在下列位置中尋找主機的公有金鑰，其中的 <ssh_host_rsa_key_name> 為主機公有金鑰的唯一名稱：

```
:  /etc/ssh/<ssh_host_rsa_key_name>.pub
```

> **Note**
>
> Amazon Redshift 只支援 RSA 金鑰。我們不支援 DSA 索引鍵。

在步驟 5 建立您的資訊清單檔案時，您會將公有金鑰的文字貼上至資訊清單檔案項目中的「公有金鑰」資料欄。

步驟 5：建立資訊清單檔案

COPY 命令可以使用 SSH 來連接至多台主機，也可以對每台主機建立多個 SSH 連線。COPY 會透過每個主機連線執行命令，然後將命令的輸出平行載入資料表。資訊清單檔案是 JSON 格式的文字檔案，供 Amazon Redshift 用來連接至主機。資訊清單檔案指定 SSH 主機端點，以及在主機上執行以將資料傳回給 Amazon Redshift 的命令。您可以選擇在每個項目中包含主機公有金鑰、登入使用者名稱及 mandatory 旗標。

在本機電腦上建立資訊清單檔案。在稍後的步驟中，您會將檔案上傳到 Amazon S3。

資訊清單檔案的格式如下所示：

```
{
   "entries": [
     {"endpoint":"<ssh_endpoint_or_IP>",
       "command": "<remote_command>",
       "mandatory":true,
       "publickey": "<public_key>",
       "username": "<host_user_name>"},
     {"endpoint":"<ssh_endpoint_or_IP>",
       "command": "<remote_command>",
```

```
        "mandatory":true,
        "publickey": "<public_key>",
        "username": "host_user_name"}
    ]
}
```

資訊清單檔案會對每個 SSH 連線包含一個 "entries" 結構。每個項目代表單一 SSH 連線。您可以對單一主機建立多個連線，也可以對多台主機建立多個連線。如上所示，欄位名稱和值都需要雙引號。不需要雙引號的唯一值為強制欄位的布林值 **true** 或 **false**。

下列描述資訊清單檔案中的欄位。

端點

主機的 URL 地址或 IP 地址。例如，"ec2-111-222-333.compute-1.amazonaws.com" 或 "22.33.44.56"

command

主機將執行的命令，以產生文字或二進位 (gzip、lzop 或 bzip2) 輸出。命令可以是使用者 "host_user_name" 有許可執行的任何命令。命令可能只是列印檔案這麼簡單，也可能查詢資料庫或啟動指令碼。輸出 (文字檔案、gzip 二進位檔案、lzop 二進位檔案，或 bzip2 二進位檔案) 必須採用 Amazon Redshift COPY 命令可擷取的格式。如需詳細資訊，請參閱 準備您的輸入資料。

publickey

(選用) 主機的公有金鑰。如果提供公有金鑰，Amazon Redshift 會使用此金鑰來識別主機。如果未提供公有金鑰，Amazon Redshift 不會嘗試識別主機。例如，如果遠端主機的公有金鑰是：`ssh-rsa AbcCbaxxx…xxxDHKJ root@amazon.com`，請在公有金鑰欄位中輸入下列文字：`AbcCbaxxx…xxxDHKJ`。

mandatory

(選用) 指出如果嘗試連線失敗，COPY 命令是否就應該失敗。預設值為 `false`。如果 Amazon Redshift 未成功建立至少一個連線，則 COPY 命令會失敗。

使用者名稱

(選用) 用來登入主機系統並執行遠端命令的使用者名稱。使用者登入名稱與步驟 2 中用來將公有金鑰新增至主機授權金鑰檔案的登入必須相同。預設的使用者名稱為 "redshift"。

下列範例顯示完整的清單檔案，可開啟對相同主機的四個連線，並透過每個連線執行不同的命令：

```
{
  "entries": [
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata1.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"},
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata2.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"},
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata3.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"},
       {"endpoint":"ec2-184-72-204-112.compute-1.amazonaws.com",
          "command": "cat loaddata4.txt",
          "mandatory":true,
          "publickey": "ec2publickeyportionoftheec2keypair",
          "username": "ec2-user"}
     ]
}
```

步驟 6：上傳資訊清單檔案至 Amazon S3 儲存貯體

上傳資訊清單檔案至 Amazon S3 儲存貯體。如果 Amazon S3 儲存貯體與 Amazon Redshift 叢集不在同一個 AWS 區域，您必須使用此選REGION項來指定資訊清單所在的 AWS 區域。如需有關建立 Amazon S3 儲存貯體及上傳檔案的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》https://docs.aws.amazon.com/AmazonS3/latest/gsg/。

步驟 7：執行 COPY 命令以載入資料

執行 COPY 命令以連線至主機，並將資料載入至 Amazon Redshift 資料表。在 COPY 命令中，指定資訊清單檔案的明確 Amazon S3 物件路徑，並包括 SSH 選項。例如

```
copy sales
from 's3://mybucket/ssh_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|'
ssh;
```

> **Note**
>
> 如果您使用自動壓縮，COPY 命令會執行兩次資料讀取，也就是會執行遠端命令兩次。第一次讀取會提供樣本進行壓縮分析，第二次讀取就實際載入資料。如果執行遠端命令兩次可能因為潛在的副作用造成問題，您應該關閉自動壓縮。若要關閉自動壓縮，請在執行 COPY 命令時將 COMPUPDATE 選項設為 OFF。如需詳細資訊，請參閱 利用自動壓縮載入資料表。

從 Amazon DynamoDB 資料表載入資料

您可以使用 COPY 命令，從單一 Amazon DynamoDB 資料表載入具有資料的資料表。

> **Important**
>
> 除非您使用REGION選項指定 Amazon DynamoDB 表格所在的 AWS 區域，否則提供資料的 Amazon DynamoDB 表必須建立在與叢集相同的 AWS 區域中。

COPY 命令會使用 Amazon Redshift 大量平行處理 (MPP) 架構，從 Amazon DynamoDB 資料表平行讀取和載入資料。您可以藉由在 Amazon Redshift 資料表上設定配送樣式來善加利用平行處理。如需詳細資訊，請參閱 使用資料分佈樣式。

> **Important**
>
> 當 COPY 命令從 Amazon DynamoDB 資料表讀取資料時，產生的資料傳輸為該資料表佈建的輸送量的一部分。

若要避免耗用過量的佈建的讀取輸送量，建議您不要從位在生產環境中的 Amazon DynamoDB 資料表載入資料。如果您的確從生產資料表載入資料，建議您將 READRATIO 選項設定為低於未使用佈建的輸送量平均百分比。較低的 READRATIO 設定將有助於減少節流問題。若要使用整個佈建的 Amazon DynamoDB 資料表輸送量，請將 READRATIO 設定為 100。

COPY 命令會使用下列規則，將擷取自 DynamoDB 資料表的項目中的屬性名稱，與現有 Amazon Redshift 資料表中的資料欄名稱比對：

- Amazon Redshift 資料表資料欄與 Amazon DynamoDB 項目屬性為不區分大小寫的比對。如果 DynamoDB 資料表中的項目包含只有大小寫差異的多個屬性，例如 Price 和 PRICE，則 COPY 命令將會失敗。

- 取決於在 COPY 命令中搭配 EMPTYASNULL 選項指定的值，不符合 Amazon DynamoDB 資料表中屬性的 Amazon Redshift 資料表資料欄會以 NULL 或空白形式載入。
- 不符合 Amazon Redshift 資料表中資料欄的 Amazon DynamoDB 屬性則會遭捨棄。屬性是在比對之前讀取，因此即使是捨棄的屬性也會使用資料表佈建的輸送量的一部分。
- 僅支援具有純量 STRING 和 NUMBER 資料類型的 Amazon DynamoDB 屬性。不支援 Amazon DynamoDB BINARY 和 SET 資料類型。如果 COPY 命令嘗試載入具有不支援資料類型的屬性，命令將會失敗。如果屬性不符合 Amazon Redshift 資料表資料欄，COPY 不會嘗試載入它，也不會引發錯誤。

COPY 命令使用下列語法來從 Amazon DynamoDB資料表載入資料：

```
copy <redshift_tablename> from 'dynamodb://<dynamodb_table_name>'
authorization
readratio '<integer>';
```

授權值是存取 Amazon DynamoDB 表格所需的 AWS 登入資料。如果這些登入資料對應於某個使用者，該使用者必須具備許可能夠 SCAN 和 DESCRIBE 要載入的 Amazon DynamoDB 資料表。

授權值提供叢集存取 Amazon DynamoDB 表所需的 AWS 授權。許可必須包含要載入之 Amazon DynamoDB 資料表的 SCAN 和 DESCRIBE。如需所需許可的相關資訊，請參閱COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可。身分驗證的偏好方法是指定 IAM_ROLE 參數，並提供 IAM 角色的 Amazon Resource Name (ARN) 所需的許可。如需詳細資訊，請參閱 角色類型存取控制。

若要使用 IAM_ROLE 參數進行驗證，*<aws-account-id>* 和 *<role-name>* 如下列語法所示。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

下列範例顯示使用 IAM 角色進行身分驗證。

```
copy favoritemovies
from 'dynamodb://ProductCatalog'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

如需其他授權選項的相關資訊，請參閱授權參數

如果想要驗證您的資料而不實際載入資料表，請使用 NOLOAD 選項搭配 COPY 命令。

下列範例會以 DynamoDB 表格中的資料載入「我的最愛影片」表格。 my-favorite-movies-table讀取活動可能耗用最多 50% 的佈建輸送量。

```
copy favoritemovies from 'dynamodb://my-favorite-movies-table'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
readratio 50;
```

為了讓輸送量達到最大，COPY 命令會跨叢集中的運算節點，從 Amazon DynamoDB 資料表平行載入資料。

具備自動壓縮的佈建輸送量

依預設，每當您指定空白的目標資料表而沒有壓縮編碼時，COPY 命令會套用自動壓縮。自動壓縮分析最初會從 Amazon DynamoDB 資料表取樣大量資料列。樣本大小是基於 COMPROWS 參數的值。預設值為每個配量 100,000 個資料列。

取樣之後，會捨棄樣本資料列並載入整個資料表。因此，許多資料列會讀取兩次。如需自動壓縮運作方式的相關資訊，請參閱利用自動壓縮載入資料表。

> **Important**
>
> 當 COPY 命令從 Amazon DynamoDB 資料表讀取資料時，包括用於取樣的資料列，產生的資料傳輸為該資料表佈建的輸送量的一部分。

從 Amazon DynamoDB 載入多位元組資料

如果資料包含非 ASCII 多位元組字元 (例如中文或斯拉夫文字元)，您必須將資料載入 VARCHAR 欄。VARCHAR 資料類型支援四位元組 UTF-8 字元，但 CHAR 資料類型只接受單位元組 ASCII 字元。您無法將五位元組或更長的字元載入 Amazon Redshift 資料表。如需 CHAR 和 VARCHAR 的相關資訊，請參閱資料類型。

驗證資料已正確載入

在載入操作完成後，查詢 STL_LOAD_COMMITS 系統資料表，確認預期的檔案已經載入。在相同的交易內執行 COPY 命令和載入驗證，使得如果載入有問題，您可以復原整個交易。

下列查詢包含 TICKIT 資料庫中全新載入之資料表的項目：

```
select query, trim(filename) as filename, curtime, status
from stl_load_commits
```

```
where filename like '%tickit%' order by query;

 query |         filename          |          curtime           | status
-------+---------------------------+----------------------------+--------
 22475 | tickit/allusers_pipe.txt  | 2013-02-08 20:58:23.274186 |      1
 22478 | tickit/venue_pipe.txt     | 2013-02-08 20:58:25.070604 |      1
 22480 | tickit/category_pipe.txt  | 2013-02-08 20:58:27.333472 |      1
 22482 | tickit/date2008_pipe.txt  | 2013-02-08 20:58:28.608305 |      1
 22485 | tickit/allevents_pipe.txt | 2013-02-08 20:58:29.99489  |      1
 22487 | tickit/listings_pipe.txt  | 2013-02-08 20:58:37.632939 |      1
 22489 | tickit/sales_tab.txt      | 2013-02-08 20:58:37.632939 |      1
(6 rows)
```

驗證輸入資料

若要驗證 Amazon S3 輸入檔案或 Amazon DynamoDB 資料表中的資料，在實際載入資料之前，請使用 NOLOAD 選項搭配 COPY 命令。使用 NOLOAD 搭配用來載入資料的相同 COPY 命令和選項。NOLOAD 會檢查所有資料的完整性，而不需將它載入至資料庫。NOLOAD 選項會顯示如果您嘗試載入資料會發生的任何錯誤。

例如，如果您指定不正確的 Amazon S3 路徑作為輸入檔案，Amazon Redshift 會顯示下列錯誤。

```
ERROR:  No such file or directory
DETAIL:
-----------------------------------------------
Amazon Redshift error:  The specified key does not exist
code:      2
context:   S3 key being read :
location:  step_scan.cpp:1883
process:   xenmaster [pid=22199]
-----------------------------------------------
```

若要對錯誤訊息進行故障診斷，請參閱載入錯誤參考。

如需使用 NOLOAD 選項的範例，請參閱具有 NOLOAD 選項的 COPY 命令。

利用自動壓縮載入資料表

主題

- 自動壓縮的運作方式

- 自動壓縮範例

您可以根據自己對資料的評估，將壓縮編碼手動套用至資料表中的資料欄。或者，您可以使用 COPY 命令，搭配使用設為 ON 的 COMPUPDATE，根據範例資料自動分析並套用壓縮。

您可以在建立和載入全新資料表時使用自動壓縮。COPY 命令會執行壓縮分析。藉由在已填入的資料表上執行 ANALYZE COMPRESSION 命令，您也可以執行壓縮分析，而不載入資料或變更資料表上的壓縮。例如，當您想要在資料表上分析壓縮以供日後使用，同時保留現有的資料定義語言 (DDL) 陳述式時，您可以執行 ANALYZE COMPRESSION。

選擇壓縮編碼時，自動壓縮會平衡整體效能。如果排序索引鍵欄位的壓縮比相同查詢中的其他欄位高出許多，限制範圍的掃描執行效果可能較差。因此，自動壓縮將跳過排序索引鍵資料欄上的資料分析階段，並保留使用者定義的編碼類型。

如果您尚未明確定義編碼類型，自動壓縮會選擇 RAW 編碼。ANALYZE COMPRESSION 有相同的行為。為了獲得最佳的查詢效能，請考慮針對排序索引鍵使用 RAW。

自動壓縮的運作方式

將 COMPUPDATE 參數設為 ON 時，每當您對空白的目標資料表執行 COPY 命令，且所有資料表資料欄都是 RAW 編碼或無編碼時，COPY 命令會套用自動壓縮。

若要對空白資料表套用自動壓縮，而不論其目前的壓縮編碼，請在執行 COPY 命令時將 COMPUPDATE 選項設為 ON。若要關閉自動壓縮，請在執行 COPY 命令時將 COMPUPDATE 選項設為 OFF。

您不可以套用自動壓縮至已包含資料的資料表。

> **Note**
>
> 自動壓縮分析需要在載入資料中有足夠的資料列 (每個配量至少 100,000 資料列)，才能產生有意義的樣本。

自動壓縮會隨著載入交易在背景中執行這些操作：

1. 從輸入檔案載入初始的資料列樣本。樣本大小是基於 COMPROWS 參數的值。預設為 100,000。
2. 為每個資料欄選擇壓縮選項。

3. 樣本資料列從資料表移除。
4. 以所選壓縮編碼重新建立資料表。
5. 使用新編碼載入和壓縮整個輸入檔案。

執行 COPY 命令之後，資料表會完整載入、壓縮和可供使用。如果您稍後載入更多資料，附加的資料列會根據現有編碼壓縮。

如果您只想要執行壓縮分析，請執行 ANALYZE COMPRESSION，這較執行完整 COPY 更有效率。然後您可以評估結果來決定是否使用自動壓縮或手動重新建立資料表。

自動壓縮僅支援 COPY 命令。另外，您可以在建立資料表時手動套用壓縮編碼。如需手動壓縮編碼的詳細資訊，請參閱使用資料欄壓縮。

自動壓縮範例

在此範例中，假設 TICKIT 資料庫包含 LISTING 資料表的複本 (名為 BIGLIST)，而您想要在載入大約 300 萬個資料列時套用自動壓縮至此資料表。

載入和自動壓縮資料表

1. 確保資料表是空白的。您僅可以套用自動壓縮至空白資料表：

```
truncate biglist;
```

2. 以單一 COPY 命令載入資料表。雖然資料表是空白的，可能已指定部分較舊的編碼。若要促使 Amazon Redshift 執行壓縮分析，請將 COMPUPDATE 參數設定為 ON。

```
copy biglist from 's3://mybucket/biglist.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' COMPUPDATE ON;
```

   因為沒有指定 COMPROWS 選項，會使用每個配量 100,000 個資料列的預設和建議樣本大小。

3. 查看 BIGLIST 資料表的新結構描述，以便檢閱自動選擇的編碼結構。

```
select "column", type, encoding
from pg_table_def where tablename = 'biglist';

     Column     |            Type             | Encoding
----------------+-----------------------------+----------
```

```
 listid         | integer                     | az64
 sellerid       | integer                     | az64
 eventid        | integer                     | az64
 dateid         | smallint                    | none
 numtickets     | smallint                    | az64
 priceperticket | numeric(8,2)                | az64
 totalprice     | numeric(8,2)                | az64
 listtime       | timestamp without time zone | az64
```

4. 驗證已載入預期的資料列數量：

```
select count(*) from biglist;

count
---------
3079952
(1 row)
```

稍後使用 COPY 或 INSERT 陳述式將資料列附加至此資料表時，會套用相同的壓縮編碼。

優化用於縮小資料表的儲存

如果您的資料表有非常少的資料欄但有非常大量的資料列，三個隱藏的中繼資料身分資料欄 (INSERT_XID、DELETE_XID、ROW_ID) 將耗用資料表不成比例的磁碟空間量。

為了最佳化隱藏資料欄的壓縮，若可能，請在單一 COPY 交易中載入資料表。如果您使用多個不同的 COPY 命令來載入資料表，INSERT_XID 資料欄將不會壓縮良好。如果您使用多個 COPY 命令，將必須執行清空操作，但它不會改善 INSERT_XID 的壓縮。

載入預設的欄位值

您可以選擇性地在您的 COPY 命令中定義資料欄清單。如果已從資料欄清單忽略資料表中的資料欄，COPY 將使用 DEFAULT 選項提供的值 (在 CREATE TABLE 命令中指定)，或如果未指定 DEFAULT 選項，則使用 NULL 來載入資料欄。

如果 COPY 嘗試將 NULL 指派給定義為 NOT NULL 的欄，COPY 命令會失敗。如需指派 DEFAULT 選項的詳細資訊，請參閱CREATE TABLE。

從 Amazon S3 上的資料檔案載入時，資料欄清單中的資料欄必須採用與資料檔案中的欄位相同的順序。如果資料檔案中的欄位在資料欄清單中沒有對應的資料欄，COPY 命令失敗。

從 Amazon DynamoDB 資料表載入時，順序並不重要。Amazon DynamoDB 屬性中不符合 Amazon Redshift 資料表中資料欄的所有欄位則會遭捨棄。

下列限制適用於使用 COPY 命令載入 DEFAULT 值至資料表時：

- 如果 IDENTITY 資料欄包括在資料欄清單中，也必須在 COPY 命令中指定 EXPLICIT_IDS 選項，否則 COPY 命令將會失敗。相同地，如果從資料欄清單省略 IDENTITY 資料欄，並且指定了 EXPLICIT_IDS 選項，則 COPY 操作將會失敗。
- 因為針對指定資料欄評估的 DEFAULT 表達式對所有載入資料列都是相同的，使用 RANDOM() 函數的 DEFAULT 表達式將指派所有資料列相同的值。
- 包含 CURRENT_DATE 或 SYSDATE 的 DEFAULT 表達式會設為目前交易的時間戳記。

如需範例，請參閱COPY 範例中的「從具有預設值的檔案載入資料」。

針對資料載入進行故障診斷

主題

- S3 ServiceException 錯誤
- 用於對資料載入進行故障診斷的系統資料表
- 多位元組字元載入錯誤
- 載入錯誤參考

本節提供識別和解決資料載入錯誤的相關資訊。

S3 ServiceException 錯誤

最常見的 s3 ServiceException 錯誤是由於格式不正確或不正確的登入資料字串、叢集和儲存貯體位於不同 AWS 區域，以及 Amazon S3 許可不足所造成。

本節提供每個類型的錯誤的故障診斷資訊。

無效的登入資料字串

如果您的登入資料字串的格式不正確，您會收到下列錯誤訊息：

```
ERROR: Invalid credentials. Must be of the format: credentials
```

```
'aws_access_key_id=<access-key-id>;aws_secret_access_key=<secret-access-key>
[;token=<temporary-session-token>]'
```

驗證登入資料字串不包含任何空格或換行符號，並且以單引號括住。

無效的存取金鑰 ID

如果您的存取金鑰 ID 不存在，您會收到下列錯誤訊息：

```
[Amazon](500310) Invalid operation: S3ServiceException:The AWS Access Key Id you
 provided does not exist in our records.
```

這通常是複製和貼上錯誤。驗證已正確輸入存取金鑰 ID。另外，如果您使用的是暫時工作階段金鑰，請檢查 token 的值是否已經設定。

無效的私密存取金鑰

如果您的私密存取金鑰不正確，您會收到下列錯誤訊息：

```
[Amazon](500310) Invalid operation: S3ServiceException:The request signature we
 calculated does not match the signature you provided.
Check your key and signing method.,Status 403,Error SignatureDoesNotMatch
```

這通常是複製和貼上錯誤。驗證已正確輸入私密存取金鑰，並且它是存取金鑰 ID 的正確金鑰。

儲存貯體位於不同區域

COPY 命令中指定的 Amazon S3 儲存貯體必須與叢集位於相同的 AWS 區域。如果您的 Amazon S3 儲存貯體和您的叢集位於不同的區域，您會收到類似以下的錯誤：

```
ERROR: S3ServiceException:The bucket you are attempting to access must be addressed
 using the specified endpoint.
```

透過在建立儲存貯體時使用 Amazon S3 管理主控台選取區域，或是在建立儲存貯體時使用 Amazon S3 API 或 CLI 來指定端點，即可以在特定區域中建立 Amazon S3 儲存貯體。如需詳細資訊，請參閱將檔案上傳到 Amazon S3。

如需 Amazon S3 區域的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的存取儲存貯體。

或者，您可以使用 REGION 選項搭配 COPY 命令來指定區域。

存取遭拒

如果使用者沒有足夠的許可，您會收到下列錯誤訊息：

```
ERROR: S3ServiceException:Access Denied,Status 403,Error AccessDenied
```

其中一個可能的原因是憑證所識別的使用者沒有 Amazon S3 儲存貯體的 LIST 和 GET 存取權。如需其他原因的資訊，請參閱《Amazon S imple Storage Service 使用者指南》中的針對 Amazon S3 中的拒絕存取 (403 禁止) 錯誤進行疑難排解。

若要了解如何管理儲存貯體的使用者存取權，請參閱《Amazon Simple Storage Service 使用者指南》中的 Amazon S3 中的 Identity and Access Management。

用於對資料載入進行故障診斷的系統資料表

下列 Amazon Redshift 系統資料表在對資料載入問題進行故障診斷時很實用：

- 查詢 STL_LOAD_ERRORS 以探索特定載入期間發生的錯誤。
- 查詢 STL_FILE_SCAN 以檢視特定檔案的載入時間或查看是否甚至讀取特定檔案。
- 查詢 STL_S3CLIENT_ERROR 來尋找從 Amazon S3 傳輸資料時所遇到錯誤的詳細資訊。

尋找和診斷載入錯誤

1. 建立檢視或定義可傳回載入錯誤詳細資訊的查詢。下列範例會將 STL_LOAD_ERRORS 資料表聯結至 STV_TBL_PERM 資料表，以將資料表 ID 與實際資料表名稱比對。

   ```
   create view loadview as
   (select distinct tbl, trim(name) as table_name, query, starttime,
   trim(filename) as input, line_number, colname, err_code,
   trim(err_reason) as reason
   from stl_load_errors sl, stv_tbl_perm sp
   where sl.tbl = sp.id);
   ```

2. 將您的 COPY 命令中的 MAXERRORS 選項設定為足夠大的值，讓 COPY 傳回關於您的資料的實用資訊。如果 COPY 遇到錯誤，錯誤訊息會引導您查詢 STL_LOAD_ERRORS 資料表以取得詳細資訊。
3. 查詢 LOADVIEW 檢視來查看錯誤詳細資訊。例如：

```
select * from loadview where table_name='venue';
```

```
  tbl   | table_name | query |         starttime
--------+------------+-------+----------------------------
 100551 | venue      | 20974 | 2013-01-29 19:05:58.365391

|     input      | line_number | colname | err_code |       reason
+----------------+-------------+-------+----------+--------------------
| venue_pipe.txt |           1 |     0 |     1214 | Delimiter not found
```

4. 在輸入檔案或載入指令碼中修正問題，根據檢視傳回的資訊。要監看的一些一般載入錯誤包括：

   - 資料表中的資料類型和輸入資料欄位中的值不符。
   - 資料表中資料欄的數量與輸入資料中的欄位數量不符。
   - 引號不符。Amazon Redshift 同時支援單引號和雙引號；不過，這些引號必須正確成對。
   - 輸入檔案中日期/時間資料的格式不正確。
   - O 輸入文件中的ut-of-range 值（用於數字列）。
   - 超出其壓縮編碼限制的資料欄的獨特值數量。

多位元組字元載入錯誤

具有 CHAR 資料類型的資料欄僅接受單位元組 UTF-8 字元，最高為位元組值 127，或 7F hex，它也是 ASCII 字元集。VARCHAR 資料欄接受多位元組 UTF-8 字元，最長四個位元組。如需詳細資訊，請參閱 字元類型。

如果您的載入資料中的資料行包含對資料欄資料類型無效的字元，COPY 會傳回錯誤並將資料列記錄在 STL_LOAD_ERRORS 系統日誌資料表，錯誤碼 1220。ERR_REASON 欄位包括無效字元的位元組序列 (十六進位)。

修正您的載入資料中無效字元的一個替代方式是在載入程序期間取代無效的字元。若要取代無效的 UTF-8 字元，請指定 ACCEPTINVCHARS 選項搭配 COPY 命令。如果設定了 ACCEPTINVCHARS 選項，則您指定的字元會取代字碼指標。如果沒有設定 ACCEPTINVCHARS 選項，Amazon Redshift 會接受這些字元作為有效的 UTF-8。如需詳細資訊，請參閱 ACCEPTINVCHARS。

下面的字碼指標清單是有效的 UTF-8，如果沒有設定 ACCEPTINVCHARS，COPY 操作不會傳回錯誤。但是，這些字碼指標不是有效的字元。您可以使用 ACCEPTINVCHARS 選項，以您指定的字元

取代字碼指標。這些字碼指標包括範圍從 0xFDD0 到 0xFDEF 的值和最多到 0x10FFFF 的值，並且以 FFFE 或 FFFF 結尾：

- 0xFFFE, 0x1FFFE, 0x2FFFE, ..., 0xFFFFE, 0x10FFFE
- 0xFFFF, 0x1FFFF, 0x2FFFF, ..., 0xFFFFF, 0x10FFFF

下列範例顯示 COPY 嘗試將 UTF-8 字元 e0 a1 c7a4 載入 CHAR 資料欄時的錯誤原因。

```
Multibyte character not supported for CHAR
(Hint: Try using  VARCHAR). Invalid char: e0 a1 c7a4
```

如果錯誤與 VARCHAR 資料類型相關，錯誤原因將包括錯誤代碼以及無效的 UTF-8 十六進位序列。下列範例顯示 COPY 嘗試將 UTF-8 a4 載入 VARCHAR 欄位時的錯誤原因。

```
String contains invalid or unsupported UTF-8 codepoints.
Bad UTF-8 hex sequence: a4 (error 3)
```

下表列出 VARCHAR 載入錯誤的描述和建議的解決方法。如果發生這些錯誤中的一個，請以有效的 UTF-8 程式碼序列取代該字元或移除該字元。

錯誤代碼	描述
1	UTF-8 位元組序列超出 VARCHAR 支援的四位元組上限。
2	UTF-8 位元組序列不完整。COPY 在字串結尾之前找不到預期數量的多位元組字元持續位元組。
3	UTF-8 單位元組字元超出範圍。開始位元組不得為 254、255 或介於 128 和 191 (含) 之間的任何字元。
4	位元組序列中結尾位元組的值超出範圍。持續位元組必須介於 128 和 191 (含) 之間。
5	UTF-8 字元保留做為代理字組。代理字組字碼指標 (U+D800 到 U+DFFF) 無效。
8	位元組序列超出 UTF-8 字碼指標上限。
9	UTF-8 位元組序列沒有相符的字碼指標。

載入錯誤參考

如果從檔案載入資料時發生任何錯誤，請查詢 STL_LOAD_ERRORS 資料表來識別錯誤，並判斷可能的說明。下表列出資料載入期間可能發生的所有錯誤代碼：

載入錯誤代碼

錯誤代碼	描述
1200	不明的剖析錯誤。聯絡支援。
1201	在輸入檔案中找不到欄位分隔符號。
1202	輸入資料包含的資料欄多於 DDL 中所定義的資料欄。
1203	輸入資料包含的資料欄少於 DDL 中所定義的資料欄。
1204	輸入資料超出資料類型可接受的範圍。
1205	日期格式無效。請參閱 DATEFORMAT 和 TIMEFORMAT 字串 以取得有效的格式。
1206	時間戳記格式無效。請參閱 DATEFORMAT 和 TIMEFORMAT 字串 以取得有效的格式。
1207	資料包含的值超出預期的範圍 0-9。
1208	FLOAT 資料類型格式錯誤。
1209	DECIMAL 資料類型格式錯誤。
1210	BOOLEAN 資料類型格式錯誤。
1211	輸入資料行未包含資料。
1212	找不到載入檔案。
1213	指定為 NOT NULL 的欄位未包含資料。
1214	找不到分隔符號。
1215	CHAR 欄位錯誤。

錯誤代碼	描述
1216	輸入行無效。
1217	識別欄值無效。
1218	使用 NULL AS '\0' 時，包含 null 結束字元的欄位 (NUL 或 UTF-8 0000) 包含了一個以上的字元。
1219	UTF-8 十六進制包含無效數字。
1220	字串包含無效或不支援的 UTF-8 字碼指標。
1221	檔案的編碼與 COPY 命令中指定的不同。
1222	整數值溢出錯誤。
1223	資料類型無效。
1224	輸入資料不是格式正確的 JSON 格式或超級資料類型。
8001	使用 MANIFEST 參數的 COPY 需要 Amazon S3 物件的完整路徑。
9005	指定的結束鍵無效。

持續從 Amazon S3 擷取檔案 (預覽)

這是預覽版本中的自動複製 (SQL 複製工作) 的發行前文件。文件和功能會隨時變更。我們建議僅在測試環境中使用此功能，不要在生產環境中使用。公開預覽將於 2024 年 7 月 31 日結束。預覽叢集將在預覽版結束的兩週後自動移除。如需預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

> **Note**
>
> 您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。 AWS 區域 會列出目前帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的建立叢集。
5. 選擇建立叢集按鈕以建立預覽叢集。
6. 當您的預覽叢集可用時，請使用 SQL 用戶端載入和查詢資料。

您的叢集必須使用名為 `preview_2023` 的預覽軌道建立。使用新叢集進行測試時，不支援將叢集還原至此軌道。自動複製功能不適用於 Amazon Redshift Serverless 工作群組。
此預覽可在下列項目中使用 AWS 區域：

- 美國東部 (俄亥俄) 區域 (us-east-2)
- 美國東部 (維吉尼亞北部) 區域 (us-east-1)
- 美國西部 (奧勒岡) 區域 (us-west-2)
- 亞太區域 (東京) 區域 (ap-northeast-1)
- 歐洲 (斯德哥爾摩) 區域 (eu-north-1)
- 歐洲 (愛爾蘭) 區域 (eu-west-1)

您可以使用 COPY JOB，將資料從儲存在 Amazon S3 中的檔案載入您的 Amazon Redshift 資料表。Amazon Redshift 會偵測新的 Amazon S3 檔案何時新增至 COPY 命令中指定的路徑。然後，COPY 命令會自動執行，您不必建立外部資料擷取管道。Amazon Redshift 會追蹤哪些檔案已載入完成。Amazon Redshift 會決定每個 COPY 命令批次處理的檔案數量。您可以在系統檢視中看到產生的 COPY 命令。

您只會定義 COPY JOB 一次。之後的執行會使用相同參數。

您可以使用 CREATE、LIST、SHOW、DROP、ALTER 和 RUN 工作的選項來管理載入操作。如需詳細資訊，請參閱 COPY JOB (預覽)。

您可以查詢系統檢視以查看 COPY JOB 狀態和進度。提供的視圖如下：

- SYS_COPY_JOB (預覽) — 目前定義的每個 COPY JOB 各包含一列。
- STL_LOAD_ERRORS — 包含來自 COPY 命令的錯誤。
- STL_LOAD_COMMITS — 包含用來對 COPY 命令資料載入進行疑難排解的資訊。
- SYS_LOAD_HISTORY — 包含 COPY 命令的詳細資訊。
- SYS_LOAD_ERROR_DETAIL — 包含 COPY 命令錯誤的詳細資訊。

若要取得由 COPY JOB 載入的檔案清單，請執行下列範例並取代 *<job_id>*：

```
SELECT job_id, job_name, data_source, copy_query,filename,status, curtime
FROM sys_copy_job copyjob
JOIN stl_load_commits loadcommit
ON copyjob.job_id = loadcommit.copy_job_id
WHERE job_id = <job_id>;
```

使用 DML 命令更新資料表

Amazon Redshift 支援標準資料處理語言 (DML) 命令 (INSERT、UPDATE 和 DELETE)，您可以使用這些命令來修改資料表中的資料列。您也可以使用 TRUNCATE 命令來執行快速的大量刪除。

> **Note**
>
> 我們強烈鼓勵您使用 COPY 命令來載入大量資料。使用個別 INSERT 陳述式填入資料表的速度可能會相當慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用 INSERT INTO ... SELECT FROM 或 CREATE TABLE AS 來改善效能。如需詳細資訊，請參閱 INSERT 或 CREATE TABLE AS。

如果您插入、更新或刪除資料表中的大量資料列 (相對於變更之前的資料列數量)，完成時請對資料表執行 ANALYZE 和 VACUUM 命令。如果隨著時間在您的應用程式中累積少量的變更，您可以排程 ANALYZE 和 VACUUM 命令定期執行。如需詳細資訊，請參閱 分析資料表 及 清空資料表。

更新和插入新資料

您可以透過使用 MERGE 命令有效地將新資料增加到現有的資料表中。建立臨時資料表，然後使用本節所述的其中一個方法從臨時資料表中更新目標資料表，即可執行合併操作。如需 MERGE 命令的相關資訊，請參閱 MERGE。

主題

合併範例 會使用名為 TICKIT 資料集的 Amazon Redshift 範例資料集。作為先決條件，您可以依照開始使用一般資料庫任務中的指示，設定 TICKIT 資料表和資料。有關範例資料集的更多詳細資訊，請參閱範例資料庫。

合併方法 1：取代現有資料列

如果您要覆寫目標資料表中的所有資料欄，執行合併最快的方式是取代現有的資料列。這只會掃描目標資料表一次，透過使用內部聯結來刪除將要更新的資料列。刪除資料列之後，這些資料列會透過臨時資料表中的單一插入操作被新資料列取代。

如果下列各項皆成立，請使用此方法：

- 您的目標資料表和您的臨時資料表包含相同的資料欄。
- 您想要將目標資料表資料欄中的所有資料取代為所有臨時資料表資料欄。
- 您將在合併中使用臨時資料表的所有資料列。

如果任一條件不適用，請使用「合併方法 2：指定資料欄清單」，而不是使用 MERGE，如下一節所示。

如果您將不會使用臨時資料表中的所有資料列，請使用 WHERE 子句來篩選 DELETE 和 INSERT 陳述式，將未變更的資料列省略。不過，如果臨時資料表中的多數資料列將不會參與合併，我們建議在個別步驟中執行 UPDATE 和 INSERT，如本節稍後所述。

合併方法 2：指定欄位清單而不使用 MERGE

使用此方法來更新目標資料表中的特定資料欄，而不覆寫整個資料列。此方法需要的時間較先前的方法更久，因為它需要額外的更新步驟，並且不會使用 MERGE 命令。如果下列任一項成立，請使用此方法：

- 並非目標資料表中的所有資料欄都是要更新的。
- 臨時資料表中的多數資料列將不會用於更新。

建立暫時的預備資料表

臨時資料表為暫時的資料表，其中保存了將用來對目標資料表進行變更的所有資料，包括更新和插入。

合併操作需要臨時資料表和目標資料表之間的聯結。若要共置聯結資料列，請將臨時資料表的分佈索引鍵設定為與目標資料表的分佈索引鍵相同的資料欄。例如，如果目標資料表使用外部索引鍵資料欄做為其分佈索引鍵，請對臨時資料表的分佈索引鍵使用相同的資料欄。如果您使用 CREATE TABLE LIKE 陳述式建立臨時資料表，臨時資料表將從父資料表繼承分佈索引鍵。如果您使用 CREATE TABLE AS 陳述式，新資料表將不會繼承分佈索引鍵。如需更多資訊，請參閱使用資料分佈樣式

如果分佈索引鍵與主索引鍵不同，並且分佈索引鍵不會隨著合併操作更新，請在分佈索引鍵資料欄上新增備援聯結述詞以啟用共置聯結。例如：

```
where target.primarykey = stage.primarykey
and target.distkey = stage.distkey
```

若要驗證查詢將使用共置聯結，請使用 EXPLAIN 執行查詢，並檢查所有聯結上的 DS_DIST_NONE。如需更多資訊，請參閱評估查詢計畫

取代現有資料列來執行合併操作

當您執行程序中詳述的合併操作時，請將建立和卸除臨時資料表以外的所有步驟置於單一交易中。如果有任何步驟失敗，您可以復原交易。使用單一交易也會減少認可的數目，如此可節省時間和資源。

取代現有資料列來執行合併操作

1. 建立臨時資料表，然後將它填入要合併的資料，如下列虛擬程式碼所示。

```
create temp table stage (like target);
```

```
insert into stage
select * from source
where source.filter = 'filter_expression';
```

2. 您可以使用 MERGE 與臨時資料表執行內部聯結，以更新目標資料表中符合臨時資料表的資料列，然後將所有剩餘的資料列插入與臨時資料表不符的目標資料表中。

   我們建議您在單一 MERGE 命令中執行更新和插入操作。

```
MERGE INTO target
USING stage [optional alias] on (target.primary_key = stage.primary_key)
WHEN MATCHED THEN
UPDATE SET col_name1 = stage.col_name1 , col_name2= stage.col_name2, col_name3 =
 {expr}
WHEN NOT MATCHED THEN
INSERT (col_name1 , col_name2, col_name3) VALUES (stage.col_name1, stage.col_name2,
 {expr});
```

3. 捨棄臨時資料表。

```
drop table stage;
```

指定欄位清單但不使用 MERGE 命令，以執行合併操作

當您執行程序中詳述的合併操作時，請將所有步驟置於單一交易中。如果有任何步驟失敗，您可以復原交易。使用單一交易也會減少認可的數目，如此可節省時間和資源。

指定資料欄清單來執行合併操作

1. 將整個操作放在單一交易區塊中。

```
begin transaction;
…
end transaction;
```

2. 建立臨時資料表，然後將它填入要合併的資料，如下列虛擬程式碼所示。

```
create temp table stage (like target);
insert into stage
select * from source
```

```
where source.filter = 'filter_expression';
```

3. 使用內部聯結搭配臨時資料表來更新目標資料表。

   - 在 UPDATE 子句中，明確列出要更新的資料欄。
   - 使用臨時資料表來執行內部聯結。
   - 如果分佈索引鍵與主索引鍵不同，並且分佈索引鍵將不會更新，請在分佈索引鍵上新增備援聯結。若要驗證查詢將使用共置聯結，請使用 EXPLAIN 執行查詢，並檢查所有聯結上的 DS_DIST_NONE。如需更多資訊，請參閱評估查詢計畫
   - 如果您的目標資料表是依時間戳記排序，請在目標資料表上新增述詞來利用受範圍限制的掃描。如需詳細資訊，請參閱 設計查詢的 Amazon Redshift 最佳實務。
   - 如果您將不會在合併中使用所有資料列，請新增子句來篩選要變更的資料列。例如，在一或多個資料欄上新增不相等篩選條件，以排除未變更的資料列。
   - 將更新、刪除和插入操作放置在單一交易區塊，如此一來若發生問題，各個項目都將能復原。

   例如：

```
begin transaction;

update target
set col1 = stage.col1,
col2 = stage.col2,
col3 = 'expression'
from stage
where target.primarykey = stage.primarykey
and target.distkey = stage.distkey
and target.col3 > 'last_update_time'
and (target.col1 != stage.col1
or target.col2 != stage.col2
or target.col3 = 'filter_expression');
```

4. 使用內部聯結搭配目標資料表從臨時資料表刪除不需要的資料列。目標資料表中的部分資料列已符合臨時資料表中的對應資料列，而其他資料列在先前的步驟中已更新。在任一情況下，插入不需要這些資料。

```
delete from stage
using target
where stage.primarykey = target.primarykey;
```

5. 從臨時資料表插入其餘的資料列。在您於步驟二用於 UPDATE 陳述式的 VALUES 子句中使用相同的資料欄清單。

```
insert into target
(select col1, col2, 'expression'
from stage);

end transaction;
```

6. 捨棄臨時資料表。

```
drop table stage;
```

合併範例

下列範例會執行合併以更新 SALES 資料表。第一個範例使用較簡易的方法，就是從目標資料表刪除，然後從臨時資料表插入所有資料列。第二個範例需要更新目標資料表中的特定資料欄，因此包括額外的更新步驟。

合併範例 會使用名為 TICKIT 資料集的 Amazon Redshift 範例資料集。作為先決條件，您可以依照開始使用一般資料庫任務指南中的指示，設定 TICKIT 資料表和資料。有關範例資料集的更多詳細資訊，請參閱範例資料庫。

範例合併資料來源

本節中的範例需要包括更新和插入的樣本資料來源。針對範例，我們將建立名為 SALES_UPDATE 的樣本資料表，其使用來自 SALES 資料表的資料。我們將以代表 12 月新銷售活動的隨機資料填入新的資料表。我們將使用 SALES_UPDATE 樣本資料表在以下的範例中建立臨時資料表。

```
-- Create a sample table as a copy of the SALES table.

create table tickit.sales_update as
select * from tickit.sales;

-- Change every fifth row to have updates.

update tickit.sales_update
set qtysold = qtysold*2,
pricepaid = pricepaid*0.8,
commission = commission*1.1
```

```
where saletime > '2008-11-30'
and mod(sellerid, 5) = 0;

-- Add some new rows to have inserts.
-- This example creates a duplicate of every fourth row.

insert into tickit.sales_update
select (salesid + 172456) as salesid, listid, sellerid, buyerid, eventid, dateid,
 qtysold, pricepaid, commission, getdate() as saletime
from tickit.sales_update
where saletime > '2008-11-30'
and mod(sellerid, 4) = 0;
```

根據相符索引鍵取代現有資料列的合併範例

下列指令碼使用 SALES_UPDATE 資料表在 SALES 資料表上執行與 12 月銷售活動新資料的合併操作。此範例會在有更新的 SALES 資料表中取代資料列。針對此範例，我們將會更新 qtysold 和 pricepaid 資料欄，但將 commission 和 saletime 維持不變。

```
MERGE into tickit.sales
USING tickit.sales_update sales_update
on ( sales.salesid = sales_update.salesid
and sales.listid = sales_update.listid
and sales_update.saletime > '2008-11-30'
and (sales.qtysold != sales_update.qtysold
or sales.pricepaid != sales_update.pricepaid))
WHEN MATCHED THEN
update SET qtysold = sales_update.qtysold,
pricepaid = sales_update.pricepaid
WHEN NOT MATCHED THEN
INSERT (salesid, listid, sellerid, buyerid, eventid, dateid, qtysold , pricepaid,
 commission, saletime)
values (sales_update.salesid, sales_update.listid, sales_update.sellerid,
 sales_update.buyerid, sales_update.eventid,
sales_update.dateid, sales_update.qtysold , sales_update.pricepaid,
 sales_update.commission, sales_update.saletime);

-- Drop the staging table.
drop table tickit.sales_update;

-- Test to see that commission and salestime were not impacted.
SELECT sales.salesid, sales.commission, sales.salestime, sales_update.commission,
 sales_update.salestime
```

```
FROM tickit.sales
INNER JOIN tickit.sales_update sales_update
ON
sales.salesid = sales_update.salesid
AND sales.listid = sales_update.listid
AND sales_update.saletime > '2008-11-30'
AND (sales.commission != sales_update.commission
OR sales.salestime != sales_update.salestime);
```

指定資料欄清單而不使用 MERGE 的合併範例

下列範例會執行合併操作，以 12 月銷售活動的新資料更新 SALES。我們需要包括更新和插入的樣本資料，以及沒有變更的資料列。針對此範例，我們想要更新 QTYSOLD 和 PRICEPAID 資料欄，例將 COMMISSION 和 SALETIME 維持不變。下列指令碼使用 SALES_UPDATE 資料表在 SALES 資料表上執行合併操作。

```
-- Create a staging table and populate it with rows from SALES_UPDATE for Dec
create temp table stagesales as select * from sales_update
where saletime > '2008-11-30';

-- Start a new transaction
begin transaction;

-- Update the target table using an inner join with the staging table
-- The join includes a redundant predicate to collocate on the distribution key -- A
 filter on saletime enables a range-restricted scan on SALES

update sales
set qtysold = stagesales.qtysold,
pricepaid = stagesales.pricepaid
from stagesales
where sales.salesid = stagesales.salesid
and sales.listid = stagesales.listid
and stagesales.saletime > '2008-11-30'
and (sales.qtysold != stagesales.qtysold
or sales.pricepaid != stagesales.pricepaid);

-- Delete matching rows from the staging table
-- using an inner join with the target table

delete from stagesales
using sales
where sales.salesid = stagesales.salesid
```

```
and sales.listid = stagesales.listid;

-- Insert the remaining rows from the staging table into the target table
insert into sales
select * from stagesales;

-- End transaction and commit
end transaction;

-- Drop the staging table
drop table stagesales;
```

執行深層複製

深層複製會使用會自動排序資料表的大量插入來重新建立和重新填入資料表。如果資料表有大型未排序的區域，深層複製較清空的速度快得多。建議您僅在可以追蹤更新時，再於深層複製操作期間進行並行更新。該程序完成後，將差異更新移到新資料表中。VACUUM 操作自動支援並行更新。

您可以選擇以下其中一個方法來建立原始資料表的複本：

- 使用原始資料表 DDL。

  如果 CREATE TABLE DDL 可供使用，則這是最快速和偏好的方法。如果建立新的資料表，您可以指定所有資料表和資料欄屬性，包括主索引鍵和外部索引鍵。您可以透過使用 SHOW TABLE 功能找到原始的 DDL。

- 使用 CREATE TABLE LIKE。

  如果原始 DDL 無法使用，您可以使用 CREATE TABLE LIKE 來重新建立原始資料表。新的資料表會繼承父資料表的編碼、分佈索引鍵、排序索引鍵和非 null 屬性。新資料表不會繼承父資料表的主索引鍵和外部索引鍵屬性，但您可以使用 ALTER TABLE 新增它們。

- 建立暫時資料表並截斷原始資料表。

  如果您必須保留父資料表的主索引鍵和外部索引鍵屬性。如果父資料表具有相依性，您可以使用 CREATE TABLE... AS (CTAS) 來建立暫時資料表。然後截斷原始資料表並從暫時資料表填入它。

  相較於使用永久資料表，使用暫時資料表可大幅改善效能，但有遺失資料的風險。暫時資料表在結束建立它所在的工作階段中時自動捨棄。TRUNCATE 會立即認可，即使它在交易區塊內。如果 TRUNCATE 成功但工作階段在後續 INSERT 完成之前關閉，則資料會遺失。如果無法接受資料遺失，請使用永久資料表。

建立資料表副本之後，您可能必須授與新資料表的存取權。您可以使用 GRANT 來定義存取權限。若要檢視並授與資料表的所有存取權限，您必須是下列其中一個角色：

- 超級使用者。
- 您想要複製之資料表的擁有者。
- 具有 ACCESS SYSTEM TABLE 權限可查看資料表權限，且具有所有相關許可授權的使用者。

此外，您可能必須針對深層複製所在的結構描述授與使用許可。如果深層複製的結構描述與原始資料表的結構描述不同，且也不是 public 結構描述，則必須授與使用許可。若要檢視並授與使用權限，您必須是下列其中一個角色：

- 超級使用者。
- 可針對深層複製結構描述授與 USES 許可的使用者。

使用原始資料表 DDL 來執行深層複製

1. (選用) 執行名為 v_generate_tbl_ddl 的指令碼來重新建立資料表 DDL。
2. 使用原始 CREATE TABLE DDL 建立資料表的複本。
3. 使用 INSERT INTO … SELECT 陳述式以來自原始資料表的資料填入該複本。
4. 檢查舊資料表上授與的權限。您可以在 SVV_RELATION_PRIVILEGES 系統檢視中看到這些許可。
5. 如有必要，請對新資料表授與舊資料表的許可。
6. 針對在原始資料表中具有權限的每個群組和使用者授與使用許可。如果您的深層複製資料表位於 public 結構描述中，或與原始資料表位於相同的結構描述中，則不需要執行此步驟。
7. 捨棄原始資料表。
8. 使用 ALTER TABLE 陳述式將複本重新命名為原始資料表名稱。

下列範例會使用名為 sample_copy 的 SAMPLE 複本，在 SAMPLE 資料表上執行深層複製。

```
--Create a copy of the original table in the sample_namespace namespace using the
 original CREATE TABLE DDL.
create table sample_namespace.sample_copy ( … );

--Populate the copy with data from the original table in the public namespace.
insert into sample_namespace.sample_copy (select * from public.sample);
```

```
--Check SVV_RELATION_PRIVILEGES for the original table's privileges.
select * from svv_relation_privileges where namespace_name = 'public' and relation_name
 = 'sample' order by identity_type, identity_id, privilege_type;

--Grant the original table's privileges to the copy table.
grant DELETE on table sample_namespace.sample_copy to group group1;
grant INSERT, UPDATE on table sample_namespace.sample_copy to group group2;
grant SELECT on table sample_namespace.sample_copy to user1;
grant INSERT, SELECT, UPDATE on table sample_namespace.sample_copy to user2;

--Grant usage permission to every group and user that has privileges in the original
 table.
grant USAGE on schema sample_namespace to group group1, group group2, user1, user2;

--Drop the original table.
drop table public.sample;

--Rename the copy table to match the original table's name.
alter table sample_namespace.sample_copy rename to sample;
```

使用 CREATE TABLE LIKE 來執行深層複製

1. 使用 CREATE TABLE LIKE 來建立新的資料表。
2. 使用 INSERT INTO … SELECT 陳述式將資料列從目前資料表複製到新資料表。
3. 檢查舊資料表上授與的權限。您可以在 SVV_RELATION_PRIVILEGES 系統檢視中看到這些許可。
4. 如有必要，請對新資料表授與舊資料表的許可。
5. 針對在原始資料表中具有權限的每個群組和使用者授與使用許可。如果您的深層複製資料表位於 public 結構描述中，或與原始資料表位於相同的結構描述中，則不需要執行此步驟。
6. 捨棄目前的資料表。
7. 使用 ALTER TABLE 陳述式將新資料表重新命名為原始資料表名稱。

下列範例會使用 CREATE TABLE LIKE 在 SAMPLE 資料表上執行深層複製。

```
--Create a copy of the original table in the sample_namespace namespace using CREATE
 TABLE LIKE.
create table sameple_namespace.sample_copy (like public.sample);
```

```
--Populate the copy with data from the original table.
insert into sample_namespace.sample_copy (select * from public.sample);

--Check SVV_RELATION_PRIVILEGES for the original table's privileges.
select * from svv_relation_privileges where namespace_name = 'public' and relation_name
 = 'sample' order by identity_type, identity_id, privilege_type;

--Grant the original table's privileges to the copy table.
grant DELETE on table sample_namespace.sample_copy to group group1;
grant INSERT, UPDATE on table sample_namespace.sample_copy to group group2;
grant SELECT on table sample_namespace.sample_copy to user1;
grant INSERT, SELECT, UPDATE on table sample_namespace.sample_copy to user2;

--Grant usage permission to every group and user that has privileges in the original
 table.
grant USAGE on schema sample_namespace to group group1, group group2, user1, user2;

--Drop the original table.
drop table public.sample;

--Rename the copy table to match the original table's name.
alter table sample_namespace.sample_copy rename to sample;
```

建立暫時資料表並截斷原始資料表來執行深層複製

1. 使用 CREATE TABLE AS 以來自原始資料表的資料列建立暫時資料表。
2. 截斷目前的資料表。
3. 使用 INSERT INTO … SELECT 陳述式將資料列從暫時資料表複製到原始資料表。
4. 捨棄暫時資料表。

下列範例會透過建立暫存資料表和截斷原始資料表，在 SALES 資料表上執行深層複製。由於原始資料表仍會保留，因此您不需要對複製資料表授與許可。

```
--Create a temp table copy using CREATE TABLE AS.
create temp table salestemp as select * from sales;

--Truncate the original table.
truncate sales;

--Copy the rows from the temporary table to the original table.
insert into sales (select * from salestemp);
```

```
--Drop the temporary table.
drop table salestemp;
```

分析資料表

ANALYZE 操作可更新查詢規劃器用來選擇最佳計畫的統計中繼資料。

在大多數情況下，您不需要明確執行 ANALYZE 命令。Amazon Redshift 會監控工作負載的變更，並在背景自動更新統計資料。此外，COPY 命令在將資料載入至空白資料表時會自動執行分析。

若要明確分析資料表或整個資料庫，請執行 ANALYZE 命令。

主題

- 自動分析
- 新資料表資料的分析
- ANALYZE 命令歷史記錄

自動分析

Amazon Redshift 會持續監控資料庫，並自動在背景中執行分析操作。為了盡量減少對系統效能的影響，自動分析會在工作負載較低期間執行。

預設會啟用自動分析。若要關閉自動分析，請修改叢集的參數群組，將 `auto_analyze` 參數設為 **`false`**。

為了減少處理時間並提高整體系統效能，對於修改程度較小的任何資料表，Amazon Redshift 會略過自動分析。

分析作業會略過具有 up-to-date 統計資料的表格。如果您在擷取、轉換和載入 (ETL) 工作流程中執行 ANALYZE，自動分析會略過有最新統計資訊的資料表。同樣地，當自動分析已更新資料表的統計資訊時，明確 ANALYZE 會略過資料表。

新資料表資料的分析

依預設，COPY 命令在將資料載入至空白資料表之後會執行 ANALYZE。設定 STATUPDATE ON，即可以強制 ANALYZE 而不論資料表是否空白。如果您指定 STATUPDATE OFF，則不會執行

ANALYZE。將 STATUPDATE 設為 ON 時，只有資料表擁有者或超級使用者可以執行 ANALYZE 命令或執行 COPY 命令。

Amazon Redshift 也會分析您使用下列命令建立的新資料表：

- CREATE TABLE AS (CTAS)
- CREATE TEMP TABLE AS
- SELECT INTO

對最初載入資料後未分析的新資料表執行查詢時，Amazon Redshift 會傳回警告訊息。在後續更新或載入之後查詢資料表時不會發生警告。如果查詢所參考的資料表尚未分析，當您在該資料表上執行 EXPLAIN 命令時，將會傳回相同的警告訊息。

每當將資料新增至非空白資料表就會大幅改變資料表大小時，您可以明確更新統計資訊。若要這麼做，請執行 ANALYZE 命令，或在 COPY 命令中使用 STATUPDATE ON 選項。若要檢視自前次 ANALYZE 以來插入或刪除的資料列數目的詳細資訊，請查詢 PG_STATISTIC_INDICATOR 系統目錄資料表。

您可以將 ANALYZE 命令的範圍指定為下列其中一項：

- 整個目前資料庫
- 單一資料表
- 單一資料表中的一或多個特定資料欄
- 可能用作查詢中述詞的資料欄

ANALYZE 命令會從資料表取得資料列的樣本，執行一些計算，並儲存所產生的資料欄統計資訊。依預設，Amazon Redshift 會為 DISTKEY 資料欄執行一個樣本階段，以及為資料表中的所有其他資料欄執行另一個樣本階段。如果您想要為資料欄的子集產生統計資料，您可以指定以逗點分隔的資料欄清單。您可以搭配 PREDICATE COLUMNS 子句執行 ANALYZE，以略過未作為述詞的欄。

ANALYZE 操作相當耗用資源，因此請僅對實際需要統計資料更新的資料表和資料欄執行。您不需要經常或按相同的排程分析所有資料表中的所有資料欄。如果資料本質上變更，請分析在下列項目中經常使用的資料欄：

- 排序和群組操作
- 聯結
- 查詢述詞

為了減少處理時間並提高整體系統效能，對於變更的資料列百分比 (取決於 analyze_threshold_percent 參數) 較低的任何資料表，Amazon Redshift 會略過 ANALYZE。依預設，分析閾值是設為 10%。您可以執行 SET 命令來變更目前工作階段的分析閾值。

較不需要頻繁分析的資料欄為代表事實和量值的那些資料欄，以及實際上從不會查詢的任何相關屬性，例如大型 VARCHAR 資料欄。例如，考慮 TICKIT 資料庫中的 LISTING 資料表。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'listing';

column         |        type        | encoding | distkey | sortkey
---------------+--------------------+----------+---------+---------
listid         | integer            | none     | t       | 1
sellerid       | integer            | none     | f       | 0
eventid        | integer            | mostly16 | f       | 0
dateid         | smallint           | none     | f       | 0
numtickets     | smallint           | mostly8  | f       | 0
priceperticket | numeric(8,2)       | bytedict | f       | 0
totalprice     | numeric(8,2)       | mostly32 | f       | 0
listtime       | timestamp with...  | none     | f       | 0
```

如果此資料表每天會載入大量的新記錄，則必須經常分析在查詢中經常用作聯結索引鍵的 LISTID 資料欄。如果 TOTALPRICE 和 LISTTIME 是經常在查詢中使用的條件，您可以在週間每天分析這些資料欄和分佈索引鍵。

```
analyze listing(listid, totalprice, listtime);
```

假設應用程式中的銷售者和事件更為靜態，而日期 ID 參考僅涵蓋兩年或三年的固定一組日期。在此情況下，這些欄的唯一值不會大幅變更。不過，每個唯一值的執行個體數目將會穩定增加。

此外，考慮相較於 TOTALPRICE 資料欄，較不常查詢 NUMTICKETS 和 PRICEPERTICKET 量值的情況。在此情況下，您可以在每個週末對整個資料表執行一次 ANALYZE 命令，以更新沒有每日分析的五個欄的統計資訊：

述詞欄位

做為指定資料欄清單的便利替代方案，您可以選擇僅分析可能用來作為述詞的資料列。執行查詢時，在聯結中使用的任何資料欄、篩選條件或群組依據子句會在系統目錄中標示為述詞資料欄。執行 ANALYZE 搭配 PREDICATE COLUMNS 子句時，分析操作只會包括符合以下條件的資料欄：

- 此資料欄已標示為述詞資料欄。

- 資料欄是分佈索引鍵。
- 資料欄是排序索引鍵的一部分。

如果沒有任一個資料表的資料欄已標示為述詞，ANALYZE 會包括所有資料欄，即使指定了 PREDICATE COLUMNS 亦然。如果沒有資料欄已標示為述詞資料欄，可能是因為尚未查詢該資料表。

當您的工作負載的查詢模式相對穩定時，您可能選擇使用 PREDICATE COLUMNS。當查詢模式變動時，由於經常使用不同的資料欄做為述詞，使用 PREDICATE COLUMNS 可能會暫時造成過時的統計資料。過時的統計資料可能導致次佳的查詢執行期計劃和較長的執行期。不過，當您下次使用 PREDICATE COLUMNS 執行 ANALYZE 時，會包含新的述詞資料欄。

若要檢視述詞資料欄的詳細資訊，請使用下列 SQL 來建立名為 PREDICATE_COLUMNS 的檢視。

```
CREATE VIEW predicate_columns AS
WITH predicate_column_info as (
SELECT ns.nspname AS schema_name, c.relname AS table_name, a.attnum as col_num,
 a.attname as col_name,
        CASE
            WHEN 10002 = s.stakind1 THEN array_to_string(stavalues1, '||')
            WHEN 10002 = s.stakind2 THEN array_to_string(stavalues2, '||')
            WHEN 10002 = s.stakind3 THEN array_to_string(stavalues3, '||')
            WHEN 10002 = s.stakind4 THEN array_to_string(stavalues4, '||')
            ELSE NULL::varchar
        END AS pred_ts
   FROM pg_statistic s
   JOIN pg_class c ON c.oid = s.starelid
   JOIN pg_namespace ns ON c.relnamespace = ns.oid
   JOIN pg_attribute a ON c.oid = a.attrelid AND a.attnum = s.staattnum)
SELECT schema_name, table_name, col_num, col_name,
       pred_ts NOT LIKE '2000-01-01%' AS is_predicate,
       CASE WHEN pred_ts NOT LIKE '2000-01-01%' THEN (split_part(pred_ts,
 '||',1))::timestamp ELSE NULL::timestamp END as first_predicate_use,
       CASE WHEN pred_ts NOT LIKE '%||2000-01-01%' THEN (split_part(pred_ts,
 '||',2))::timestamp ELSE NULL::timestamp END as last_analyze
FROM predicate_column_info;
```

假設您對 LISTING 資料表執行下列查詢。請注意，聯結、篩選條件和群組依據子句中會使用 LISTID、LISTTIME 和 EVENTID。

```
select s.buyerid,l.eventid, sum(l.totalprice)
```

```
from listing l
join sales s on l.listid = s.listid
where l.listtime > '2008-12-01'
group by l.eventid, s.buyerid;
```

查詢 PREDICATE_COLUMNS 檢視時，如下列範例所示，您會看到 LISTID、EVENTID 和 LISTTIME 都標示為述詞欄。

```
select * from predicate_columns
where table_name = 'listing';
```

```
schema_name | table_name | col_num | col_name       | is_predicate |
 first_predicate_use | last_analyze
------------+------------+---------+----------------+--------------
+---------------------+---------------------
public      | listing    |       1 | listid         | true         | 2017-05-05
 19:27:59 | 2017-05-03 18:27:41
public      | listing    |       2 | sellerid       | false        |
  | 2017-05-03 18:27:41
public      | listing    |       3 | eventid        | true         | 2017-05-16
 20:54:32 | 2017-05-03 18:27:41
public      | listing    |       4 | dateid         | false        |
  | 2017-05-03 18:27:41
public      | listing    |       5 | numtickets     | false        |
  | 2017-05-03 18:27:41
public      | listing    |       6 | priceperticket | false        |
  | 2017-05-03 18:27:41
public      | listing    |       7 | totalprice     | false        |
  | 2017-05-03 18:27:41
public      | listing    |       8 | listtime       | true         | 2017-05-16
 20:54:32 | 2017-05-03 18:27:41
```

讓統計資料保持在最新狀態，可讓查詢規劃工具選擇最佳計劃，進而改善查詢效能。Amazon Redshift 會在背景自動重新整理統計資料，您也可以明確執行 ANALYZE 命令。如果您選擇明確執行 ANALYZE，請執行下列操作：

- 執行查詢之前執行 ANALYZE 命令。
- 在每個定期載入或更新週期結束時，例行地在資料庫上執行 ANALYZE 命令。
- 在您建立的任何新資料表和經歷大幅變更的任何現有資料表或資料欄上執行 ANALYZE 命令。

- 根據在查詢中的用途和變更的傾向，考慮以不同的排程為不同類型的資料表和資料欄執行 ANALYZE 操作。
- 為了節省時間和叢集資源，執行 ANALYZE 時請使用 PREDICATE COLUMNS 子句。

將快照還原到佈建的叢集或無伺服器命名空間之後，以及在恢復暫停的佈建叢集之後，您都無須明確執行 ANALYZE 命令。在這些情況下，Amazon Redshift 會保留系統資料表資訊，因此不需要手動執行 ANALYZE 命令。Amazon Redshift 將根據需求繼續執行自動分析操作。

分析作業會略過具有 up-to-date 統計資料的表格。如果您在擷取、轉換和載入 (ETL) 工作流程中執行 ANALYZE，自動分析會略過有最新統計資訊的資料表。同樣地，當自動分析已更新資料表的統計資訊時，明確 ANALYZE 會略過資料表。

ANALYZE 命令歷史記錄

知道上次對資料表或資料庫執行 ANALYZE 命令的時間很實用。執行 ANALYZE 命令時，Amazon Redshift 會執行看起來類似以下的多個查詢：

```
padb_fetch_sample: select * from table_name
```

查詢 STL_ANALYZE 來檢視分析操作的記錄。如果 Amazon Redshift 使用自動分析來分析資料表，is_background 欄會設為 t (true)。否則會設為 f (false)。下列範例聯結 STV_TBL_PERM，以顯示資料表名稱和執行期詳細資訊。

```
select distinct a.xid, trim(t.name) as name, a.status, a.rows, a.modified_rows,
 a.starttime, a.endtime
from stl_analyze a
join stv_tbl_perm t  on t.id=a.table_id
where name = 'users'
order by starttime;

xid    | name  | status          | rows  | modified_rows | starttime           |
 endtime
-------+-------+-----------------+-------+---------------+---------------------
+---------------------
  1582 | users | Full            | 49990 |         49990 | 2016-09-22 22:02:23 |
 2016-09-22 22:02:28
244287 | users | Full            | 24992 |         74988 | 2016-10-04 22:50:58 |
 2016-10-04 22:51:01
```

```
244712 | users | Full            | 49984 |         24992 | 2016-10-04 22:56:07 |
 2016-10-04 22:56:07
245071 | users | Skipped         | 49984 |             0 | 2016-10-04 22:58:17 |
 2016-10-04 22:58:17
245439 | users | Skipped         | 49984 |          1982 | 2016-10-04 23:00:13 |
 2016-10-04 23:00:13
(5 rows)
```

或者，您可以執行更複雜的查詢，傳回在包括 ANALYZE 命令的每個完成的交易中執行的所有陳述式：

```
select xid, to_char(starttime, 'HH24:MM:SS.MS') as starttime,
datediff(sec,starttime,endtime ) as secs, substring(text, 1, 40)
from svl_statementtext
where sequence = 0
and xid in (select xid from svl_statementtext s where s.text like 'padb_fetch_sample
%' )
order by xid desc, starttime;

xid  |  starttime   | secs |                  substring
-----+--------------+------+-------------------------------------------
1338 | 12:04:28.511 |    4 | Analyze date
1338 | 12:04:28.511 |    1 | padb_fetch_sample: select count(*) from
1338 | 12:04:29.443 |    2 | padb_fetch_sample: select * from date
1338 | 12:04:31.456 |    1 | padb_fetch_sample: select * from date
1337 | 12:04:24.388 |    1 | padb_fetch_sample: select count(*) from
1337 | 12:04:24.388 |    4 | Analyze sales
1337 | 12:04:25.322 |    2 | padb_fetch_sample: select * from sales
1337 | 12:04:27.363 |    1 | padb_fetch_sample: select * from sales
...
```

清空資料表

Amazon Redshift 可以在背景中自動排序及在資料表上執行 VACUUM DELETE 操作。如要在載入或一系列的累加式更新之後清理資料表，您也可以對整個資料庫或個別資料表執行 VACUUM 命令。

> **Note**
>
> 只有具有必要資料表權限的使用者才能有效地清除資料表。若 VACUUM 執行時沒有必要的資料表權限，操作仍會成功完成，但不會有任何作用。如需有效執行 VACUUM 的有效資料表權限清單，請參閱 VACUUM。

因此，我們建議視需要清空個別資料表。我們也建議使用此方法，因為清空整個資料庫可能會是相當耗費資源的操作。

自動資料表排序

Amazon Redshift 會在背景中自動排序資料，以依據其排序索引鍵的順序維護資料表資料。Amazon Redshift 會追蹤您的掃描查詢，以判斷資料表中的哪些區段適合排序。

取決於系統上的載入，Amazon Redshift 會自動啟動排序。這種自動排序能減少執行 VACUUM 命令以將資料保持在排序索引鍵順序的需求。如果您需要資料完全依照排序索引鍵的順序排序 (例如在大型資料載入後)，您仍然可以手動執行 VACUUM 命令。如要判斷您的資料表是否能從執行 VACUUM SORT 中獲益，請監控 SVV_TABLE_INFO 中的 `vacuum_sort_benefit` 資料行。

Amazon Redshift 追蹤會掃描每個資料表上使用排序索引鍵的查詢。Amazon Redshift 會估計掃描及篩選每個資料表資料獲得改善的最大百分比 (如果資料表已完全排序的話)。您可以在 SVV_TABLE_INFO 中的 `vacuum_sort_benefit` 資料行中看見此估計。您可以使用此資料行搭配 `unsorted` 資料行，來判斷查詢何時能從對資料表手動執行 VACUUM SORT 中獲益。`unsorted` 資料行反映了資料表的實體排序順序。`vacuum_sort_benefit` 資料行指定了手動執行 VACUUM SORT 以排序資料表所會造成的影響。

例如，考量以下查詢：

```
select "table", unsorted,vacuum_sort_benefit from svv_table_info order by 1;
```

```
 table | unsorted | vacuum_sort_benefit
-------+----------+---------------------
 sales |    85.71 |                5.00
 event |    45.24 |               67.00
```

針對 “sales” 資料表，即使資料表約 86% 是實體未排序的，因 86% 未排序而造成的查詢效能影響也只有 5%。這可能是因為查詢只存取了資料表的一小部分，或是存取資料表的查詢非常少。針對 “event” 資料表，該資料表約 45% 是實體未排序的。但 67% 的查詢效能影響指出查詢可能存取了資料表的較大部分，或是存取資料表的查詢數相當龐大。“event” 資料表可能會從執行 VACUUM SORT 中獲益。

自動清空刪除

當您執行刪除時，資料列會標示為要刪除，但不會移除。Amazon Redshift 會根據資料庫資料表中已刪除的資料列數，在背景中自動執行 VACUUM DELETE 操作。Amazon Redshift 會將 VACUUM DELETE 排程在負載降低的期間執行，並在高負載期間暫停操作。

主題

VACUUM 頻率

您應該根據需求時常進行清空，才能保有一致的查詢效能。判斷執行您的 VACUUM 命令的頻率時，請考慮這些因素：

- 在您預期叢集上只有最少量活動的時段期間，例如傍晚或指定的資料庫管理時段期間執行 VACUUM。
- 在維護時段外執行 VACUUM 命令。如需詳細資訊，請參閱排程維護時段。
- 大量未排序的區域會造成較長的清空時間。如果您延遲清空，清空會耗費更長時間，因為必須重新整理更多資料。
- VACUUM 為 I/O 密集的操作，因此，清空完成所需的時間愈長，對您的叢集執行的並行查詢和其他資料庫操作造成的影響愈大。
- 對於使用交錯排序的資料表，VACUUM 需要較長的時間。如要評估是否必須重新排序交錯的資料表，請查詢 SVV_INTERLEAVED_COLUMNS 檢視。

排序階段和合併階段

Amazon Redshift 會在兩個階段中執行清空操作：首先，排序未排序區域中的資料列，然後在必要時，將資料表尾端新排序的資料列與現有資料列合併。清空大型資料表時，清空操作會以一系列的步驟進行，其中包含遞增排序，接著是合併。如果操作失敗或如果 Amazon Redshift 在清空期間離線，部分清空的資料表或資料庫將處於一致的狀態，但您將必須手動重新開始清空操作。遞增排序會遺失，但不

需要再次清空失敗之前認可的合併資料列。如果未排序的區域很大，損失的時間可能會很可觀。如需排序和合併階段的相關資訊，請參閱管理合併資料列的數量。

使用者可以在清空資料表時加以存取。在清空資料表時您可以執行查詢和寫入操作，但是當 DML 和清空並行執行時，這兩項可能都需要較長時間。如果您在清空期間執行 UPDATE 和 DELETE 陳述式，系統效能可能會降低。遞增合併會暫時封鎖並行 UPDATE 和 DELETE 操作，因此 UPDATE 和 DELETE 操作會在受影響的資料表暫時封鎖遞增合併步驟。DDL 操作，例如 ALTER TABLE，會遭到封鎖，直到資料表的清空操作完成為止。

> **Note**
>
> VACUUM 的各種修飾詞都會控制其運作方式。您可以根據當前需求，使用這些修飾詞來制定清空操作。例如，使用 VACUUM RECLUSTER 可透過不執行完整合併操作來縮短清空操作。如需詳細資訊，請參閱 VACUUM。

清空閾值

根據預設，若資料表中超過 95% 的資料列已排序，則 VACUUM 會略過資料表的排序階段。略過排序階段可大幅改善 VACUUM 的效能。若要變更單一資料表的預設排序閾值，在執行 VACUUM 命令時，請包含資料表名稱和 TO 閾值 PERCENT 參數。

清空類型

如需不同清空類型的詳細資訊，請參閱 VACUUM。

管理清空時間

取決於您的資料的本質，我們建議遵循本節中的做法來將清空時間最小化。

主題

- 決定是否重建索引
- 管理未排序區域的大小
- 管理合併資料列的數量
- 以排序索引鍵順序載入資料
- 使用時間序列資料表

決定是否重建索引

您通常可以使用交錯的排序樣式大幅改善查詢效能，但如果排序索引鍵資料欄中值的配送變更，效能可能隨著時間降級。

當您一開始使用 COPY 或 CREATE TABLE AS 載入空白的交錯資料表時，Amazon Redshift 會自動建立交錯的索引。如果您最初使用 INSERT 載入交錯的資料表，您之後必須執行 VACUUM REINDEX，才能初始化交錯的索引。

隨著時間，在您新增具有新排序索引鍵值的資料列時，如果排序索引鍵資料欄中值的配送變更，效能可能降級。如果您的新資料列主要在現有排序索引鍵值的範圍內，則不需要重建索引。執行 VACUUM SORT ONLY 或 VACUUM FULL 來還原排序順序。

查詢引擎可以使用排序順序來有效地選取處理查詢需要掃描的資料區塊。針對交錯的排序，Amazon Redshift 會分析排序索引鍵資料欄值，以判斷最佳排序順序。如果隨著資料列新增，索引鍵值的配送變更或偏移，排序策略將不再最佳，而排序的效能優勢將會降級。若要重新分析排序索引鍵配送您可以執行 VACUUM REINDEX。重建索引操作相當耗時，因此，若要決定資料表是否將可從重建索引獲益，請查詢 SVV_INTERLEAVED_COLUMNS 檢視。

例如，下列查詢顯示使用交錯排序索引鍵的資料表的詳細資訊。

```
select tbl as tbl_id, stv_tbl_perm.name as table_name,
col, interleaved_skew, last_reindex
from svv_interleaved_columns, stv_tbl_perm
where svv_interleaved_columns.tbl = stv_tbl_perm.id
and interleaved_skew is not null;

 tbl_id | table_name | col | interleaved_skew | last_reindex
--------+------------+-----+------------------+---------------------
 100048 | customer   |   0 |             3.65 | 2015-04-22 22:05:45
 100068 | lineorder  |   1 |             2.65 | 2015-04-22 22:05:45
 100072 | part       |   0 |             1.65 | 2015-04-22 22:05:45
 100077 | supplier   |   1 |             1.00 | 2015-04-22 22:05:45
(4 rows)
```

interleaved_skew 的值為比率，指出偏移量。值為 1 表示沒有偏移。如果偏移大於 1.4，VACUUM REINDEX 一般將會改善效能，除非偏移本來就在基礎設定中。

您可以使用 last_reindex 中的日期值來判斷自上次重建索引後經過的時間。

管理未排序區域的大小

載入大量新資料至已包含資料的資料表，或當您不隨著例行維護操作清空資料表時，未排序的區域會成長。若要避免長時間執行清空操作，請使用下列做法：

- 根據定期排程執行清空操作。

  如果您以小型增量載入您的資料表 (例如代表資料表中資料列總數的每日更新)，定期執行 VACUUM 將有助於確保個別清空操作快速進行。

- 先執行最大型的載入。

  如果您需要使用多個 COPY 操作載入新資料表，請先執行最大的載入。執行初始載入至新的或截斷的資料表時，所有資料會直接載入至排序的區域，因此不需要清空。

- 截斷資料表，而非刪除所有資料列。

  從資料表刪除資料列不會回收該資料列佔用的空間，直到您執行清空操作為止；不過，截斷資料表會清空資料表和回收磁碟空間，因此不需要清空。或者，捨棄資料表和重新建立它。

- 截斷或捨棄測試資料表。

  如果您因測試目的載入少量的資料列至資料表，在完成之前請勿刪除資料列。而是截斷資料表，並在後續生產載入操作中重新載入這些資料列。

- 執行深層複製。

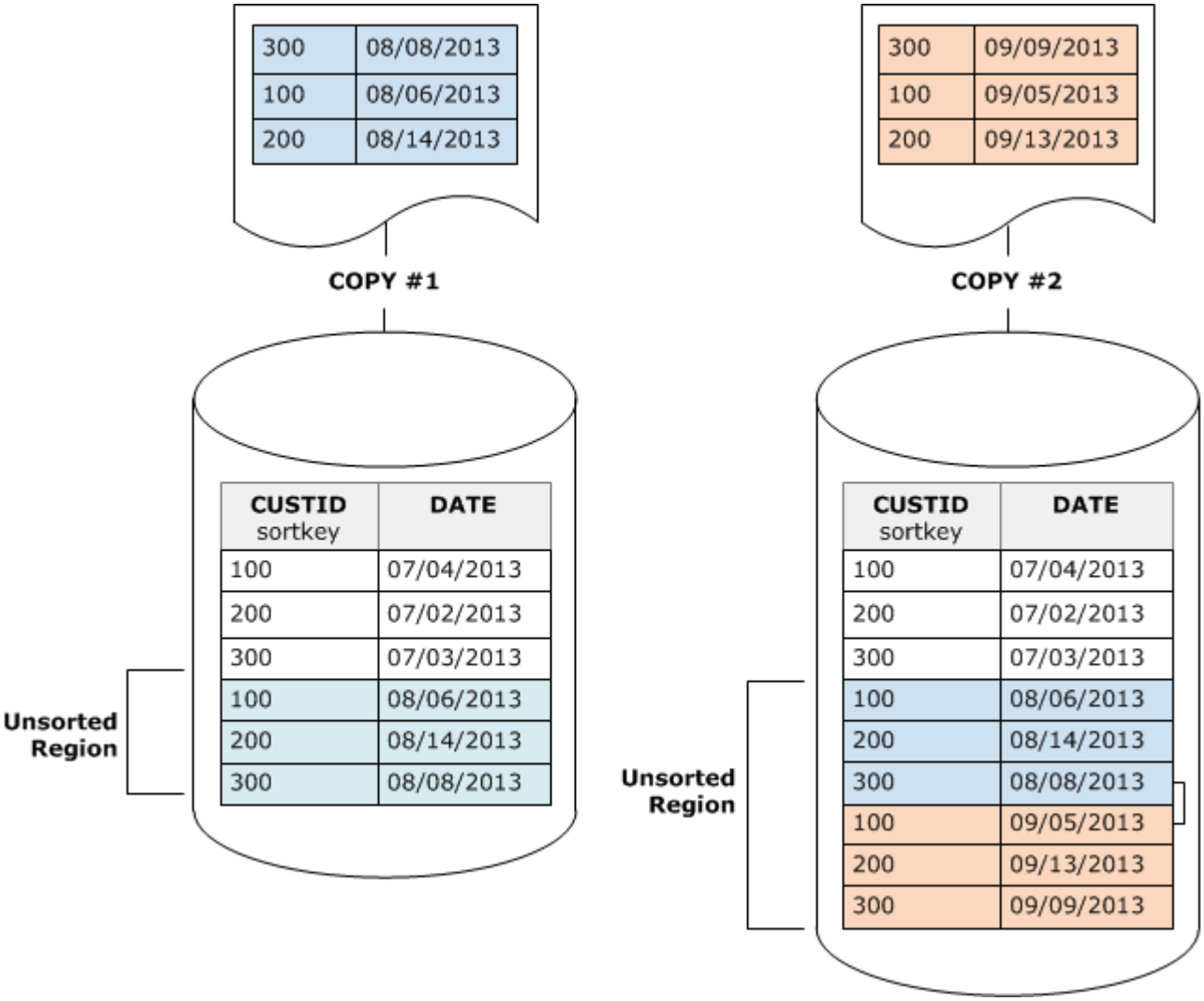
  如果使用複合排序索引鍵資料表的資料表有大型未排序的區域，深層複製較清空的速度快得多。深層複製會使用會自動重新排序資料表的大量插入來重新建立和重新填入資料表。如果資料表有大型未排序的區域，深層複製較清空的速度快得多。取捨是您不可以在深層複製操作期間進行並行更新，但清空期間卻可以。如需詳細資訊，請參閱 設計查詢的 Amazon Redshift 最佳實務。

管理合併資料列的數量

如果清空操作必須將新資料列合併至資料表的排序區域，清空所需的時間將隨著資料表變大而增加。您可以透過減少必須合併的資料列數量來改善清空效能。

在清空之前，資料表在資料表的標頭中包含排序的區域，接著是未排序的區域，它會在資料列新增或更新時成長。透過 COPY 操作新增一組資料列時，在新增至資料表尾端的未排序區域時，新的一組資料列會依排序索引鍵排序。新資料列會在其自己的集合內排序，但不會在未排序的區域內排序。

下圖說明兩個後續的 COPY 操作之後的未排序區域，其中的排序索引鍵為 CUSTID。為求簡化，此範例顯示複合排序索引鍵，但相同的原則適用於交錯的排序索引鍵，除了未排序區域的影響大於交錯的資料表。



清空會以兩個階段還原資料表的排序順序：

1. 將未排序的區域排序為新排序的區域。

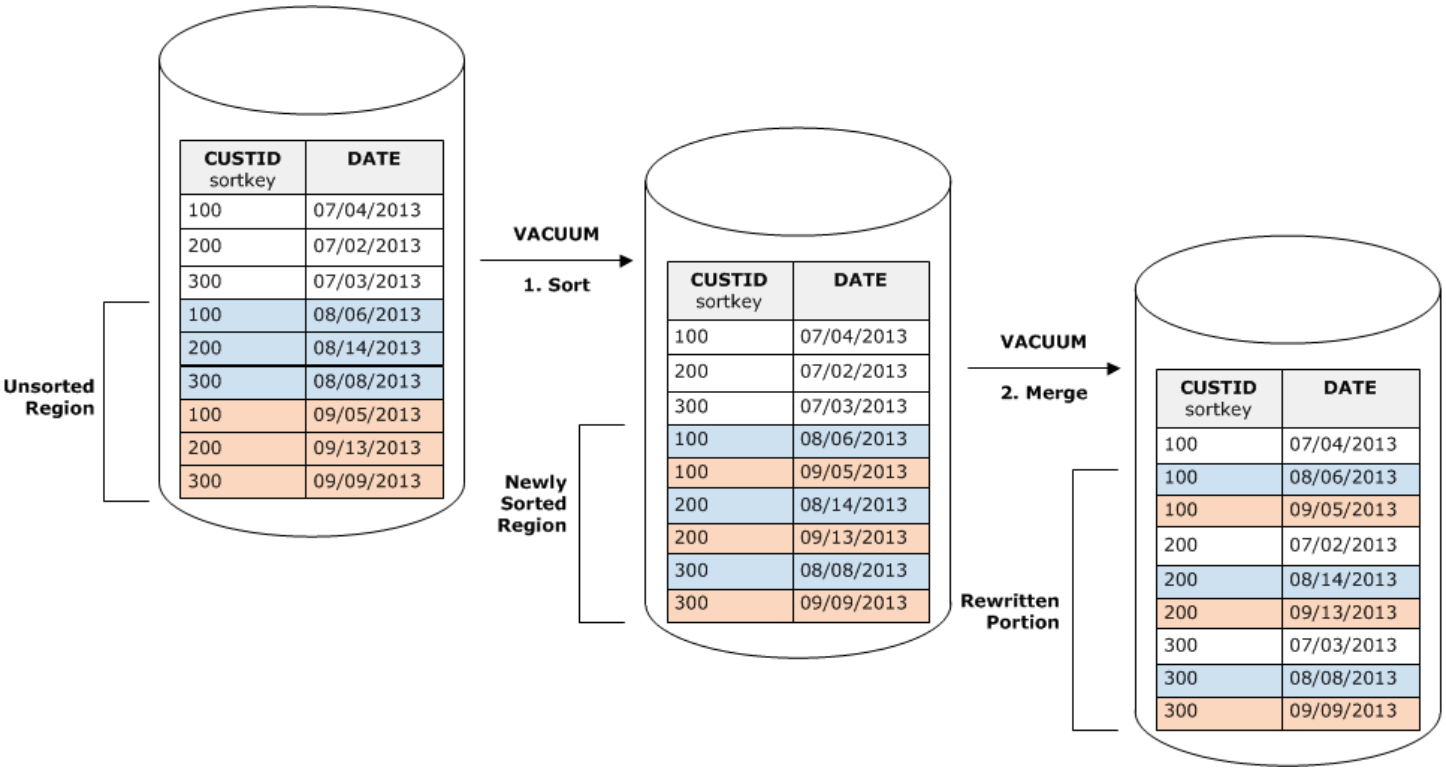
   第一個階段相當經濟實惠，因為只會重新寫入未排序的區域。如果新排序區域的排序索引鍵值的範圍高於現有範圍，則只需要重新寫入新資料列，清空即完成。例如，如果排序的區域包含 ID 值 1 到 500，而和後續的複製操作會新增大於 500 的索引鍵值，那麼只必須重新寫入未排序的區域。

2. 將新排序的區域與先前排序的區域合併。

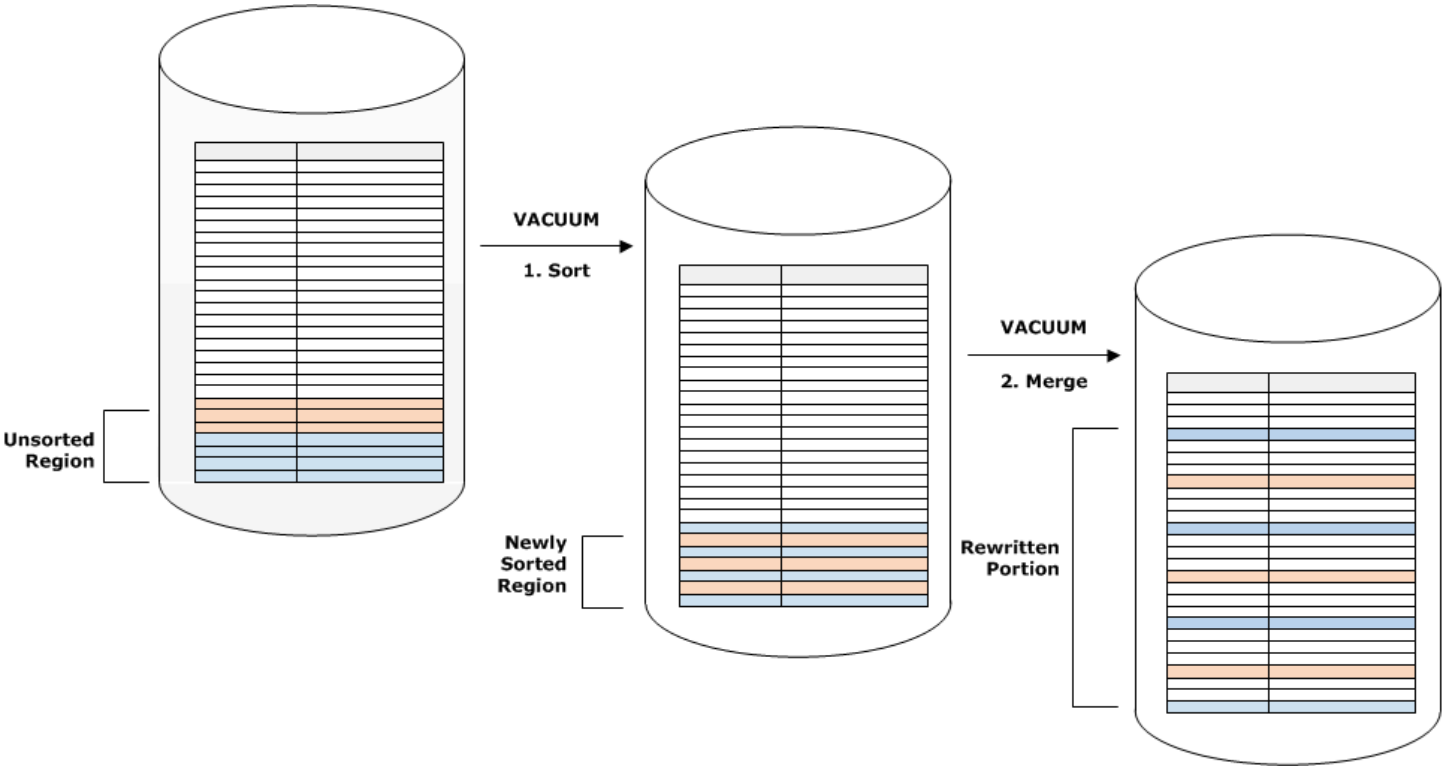
如果新排序區域中的索引鍵與排序的區域中的索引鍵重疊，那麼 VACUUM 必須合併資料列。從新排序區域 (最下方的排序索引鍵) 的開頭開始，清空會從將合併的資料列從先前排序的區域和新排序的區域寫入至新的一組區塊。

新排序索引鍵範圍與現有排序索引鍵重疊的程度，決定需要重新寫入先前排序區域的程度。如果未排序的索引鍵分散在現有排序範圍中，清空可能需要重新寫入資料表的現有部分。

下表顯示清空如何排序和合併新增至 CUSTID 為排序索引鍵的資料表的資料列。因為每個複製操作會新增一組新資料列，其具有的索引鍵值與現有索引鍵重疊，幾乎必須重新寫入整個資料表。此表顯示單一的排序和合併，但在實務上，大型的清空包含一系列的遞增排序和合併步驟。



如果一組新資料列中排序索引鍵的範圍與現有索引鍵的範圍重疊，合併階段的成本會隨著資料表成長，繼續按比例成長至資料表大小，同時排序階段的成本會維持與未排序區域的大小成比例。在這類情況下，合併階段的成本會使得排序階段的成本相形見絀，如下表所示。



若要判斷資料表重新合併的比例，請在清空操作完成之後查詢 SVV_VACUUM_SUMMARY。下列查詢顯示在 CUSTSALES 隨著時間變大時，六個後續清空的效果。

```
select * from svv_vacuum_summary
where table_name = 'custsales';

 table_name | xid  | sort_      | merge_     | elapsed_   | row_  | sortedrow_ | block_
  | max_merge_
            |      | partitions | increments | time       | delta | delta      | delta
  | partitions
 -----------+------+------------+------------+------------+-------+------------
+---------+---------------
  custsales | 7072 |          3 |          2 |  143918314 |     0 |   88297472 |   1524
  |      47
  custsales | 7122 |          3 |          3 |  164157882 |     0 |   88297472 |    772
  |      47
  custsales | 7212 |          3 |          4 |  187433171 |     0 |   88297472 |    767
  |      47
  custsales | 7289 |          3 |          4 |  255482945 |     0 |   88297472 |    770
  |      47
  custsales | 7420 |          3 |          5 |  316583833 |     0 |   88297472 |    769
  |      47
```

```
 custsales | 9007 |          3 |          6 |  306685472 |     0 |   88297472 |    772
 |      47
(6 rows)
```

merge_increments 資料欄提供針對每個清空操作所合併資料量的指示。如果合併增量的數目超過資料表大小成長比例後續清空的增加，即表示因為現有和新排序的區域重疊，每個清空操作會重新合併資料表中增加數量的資料列。

以排序索引鍵順序載入資料

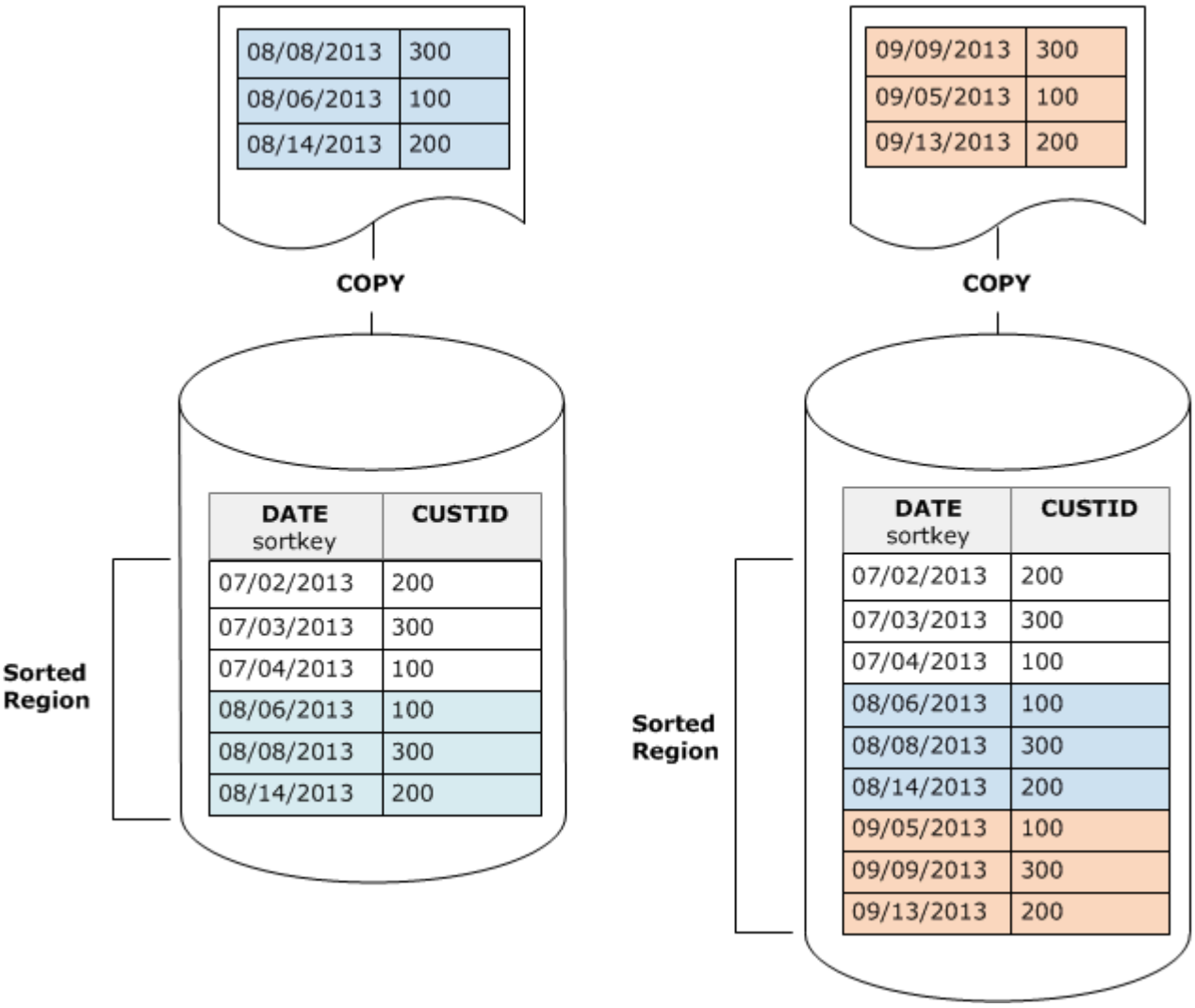
如果使用 COPY 命令以排序索引鍵順序載入您的資料，您可能會降低或甚至免除清空的需求。

當以下各項成立時，COPY 會自動新增新資料列至資料表的排序區域：

- 資料表使用複合排序索引鍵搭配僅一個排序資料欄。
- 排序資料欄為 NOT NULL。
- 資料表完全經過排序或為空白。
- 所有新資料列的排序順序高於現有資料列，包括標記進行刪除的資料列。在此執行個體中，Amazon Redshift 會使用排序索引鍵的前八個位元組來判斷排序順序。

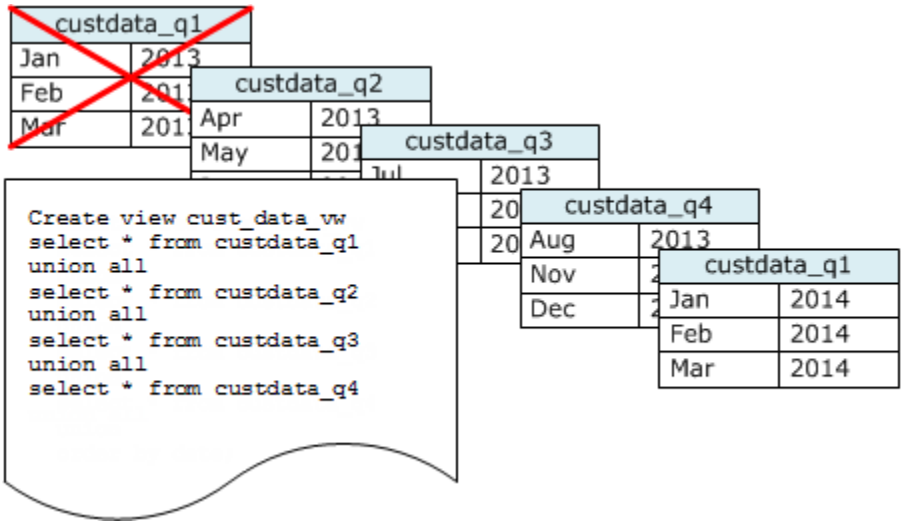
例如，假設您有一個使用客戶 ID 和時間記錄客戶事件的資料表。如果您依客戶 ID 排序，遞增載入新增的新資料列的排序索引鍵範圍很可能會與現有範圍重疊，如先前的範例所示，導致代價高昂的清空操作。

如果您將您的排序索引鍵設為時間戳記資料欄，您的新資料列將以排序順序附加在資料表結尾，如下表所示，降低或甚至消除清空的需求。



使用時間序列資料表

如果您維護資料一段時間，使用一系列資料表，如下圖所述。



每次您新增一組資料即建立新的資料表，然後刪除系列中最舊的資料表。您會獲得雙重優勢：

- 避免刪除資料列的增加成本，因為 DROP TABLE 操作較大量 DELETE 來的更有效。
- 如果資料表是依時間戳記排序，則不需要清空。如果每個資料表包含一個月的資料，清空將至少必須重新寫入一個月的資料量，即使資料表未依時間戳記排序。

您可以建立 UNION ALL 檢視，供隱藏資料儲存在多個資料表事實的報告查詢使用。如果查詢會依排序索引鍵篩選，查詢規劃器可以有效地略過所有未使用的資料表。UNION ALL 對於其他類型的查詢可能效率較低，因此您應該評估使用該資料表的所有查詢內容中的查詢效能。

管理並行寫入操作

主題

- 可序列化隔離
- 寫入和讀/寫操作
- 並行寫入範例

Amazon Redshift 允許在遞增載入或修改資料表時加以讀取。

在一些傳統資料資料倉儲和商業智慧應用程式中，只有在每夜載入完成時，資料庫才可供使用者使用。在這類情況下，當分析查詢執行和報告產生時的一般工作時段期間不允許進行更新；不過，應用程式數目的增加會長時間或甚至整天保持在線上，使得載入時段的概念過時。

Amazon Redshift 藉由允許在遞增載入或修改資料表時加以讀取，來支援這些類型的應用程式。查詢只會看到最新認可的資料版本或快照，而非等候認可的下一個版本。如果您想要特定查詢等候來自另一個寫入操作的認可，您必須據以排程。

下列主題說明一些牽涉交易、資料庫快照、更新和並行行為的重要概念和使用案例。

可序列化隔離

有些應用程式需要的不僅僅是並行查詢和載入，還有並行寫入多個資料表或相同的資料表的能力。在此情況下，並行表示重疊，而不是排程在完全相同的時間執行。如果第二個交易在第一個認可之前開始，則會把兩個交易會視為並行。並行操作可能源自受相同使用者或不同使用者控制的不同工作階段。

> **Note**
>
> Amazon Redshift 支援預設的自動遞交行為，其中每個分開執行的 SQL 命令都會個別遞交。如果您在交易區塊中含括一組命令 (由 BEGIN 和 結束 陳述式定義)，該區塊會以一個交易的形式認可，因此您可以在必要時加以復原。這種行為的例外是 TRUNCATE 和 VACUUM 命令，這兩種命令會自動遞交所有目前交易中未完成的變更。
> 某些 SQL 用戶端會自動發出 BEGIN 和 COMMIT 命令，讓用戶端控制要將陳述式群組做為交易執行，還是將每個個別的陳述式做為單獨的交易執行。請查看您正在使用的界面文件。例如，使用 Amazon Redshift JDBC 驅動程式時，附帶其中包含多個 (以分號區隔) SQL 命令查詢字串的 JDBC `PreparedStatement` 會將所有陳述式做為單一交易執行。相反地，如果您使用 SQL Workbench/J 並設定 AUTO COMMIT ON，則當您執行多個陳述式時，每個陳述式都會做為單獨的交易執行。

Amazon Redshift 中是使用資料表上的寫入鎖定及可序列化隔離的原則，以保護的方式支援並行寫入操作。可序列化隔離可維持對資料表執行的交易對該資料表執行的唯一交易的錯覺。例如，兩個並行執行的交易，T1 和 T2，必須產生與下列至少一項相同的結果：

- T1 和 T2 會依照該順序循序執行。
- T2 和 T1 會依照該順序循序執行。

並行交易彼此看不見對方；它們無法偵測彼此的變更。每個並行交易將在交易開始時建立資料庫的快照。資料庫快照會在多數 SELECT 陳述式、DML 命令 (例如 COPY、DELETE、INSERT、UPDATE 和 TRUNCATE)，以及下列 DDL 命令的第一個出現的交易內建立：

- ALTER TABLE (用來新增或捨棄資料欄)

- CREATE TABLE
- DROP TABLE
- TRUNCATE TABLE

如果並行交易的任何序列執行會產生與其並行執行相同的結果，這些交易會被視為「可序列化」並且可安全地執行。如果這些交易中沒有任何序列執行會產生相同的結果，系統會停止和還原所執行陳述式可能破壞序列化可行性的交易。

系統目錄資料表 (PG) 和其他 Amazon Redshift 系統資料表 (STL 和 STV) 不會在交易中鎖定。因此，DDL 和 TRUNCATE 操作所產生的資料庫物件變更，都會在認可至任何並行交易時顯示出來。

例如，假設當兩個並行交易 T1 和 T2 開始時，資料表 A 存在於資料庫。假設 T2 從 PG_TABLES 目錄資料表中進行選取，傳回了資料表清單。然後 T1 捨棄資料表 A 並遞交，然後 T2 再次列出資料表。資料表 A 現在不會再出現在清單中。如果 T2 嘗試查詢已捨棄的資料表，Amazon Redshift 會傳回「關聯不存在」錯誤。傳回資料表的清單至 T2 或檢查資料表 A 存在的類別查詢，不會受限於與對使用者資料表執行的相同隔離規則。

對這些資料表更新的交易會在讀取已認可隔離模式中執行。PG 字首類別資料表不支援快照隔離。

系統資料表和類別資料表的可序列化隔離

資料庫快照也會在參考使用者建立的資料表或 Amazon Redshift 系統資料表 (STL 或 STV) 的任何 SELECT 查詢的交易中建立。不參考任何資料表的 SELECT 查詢不會建立新的交易資料庫快照。僅在系統目錄資料表 (PG) 上運作的 INSERT、DELETE 和 UPDATE 陳述式也不會建立新的交易資料庫快照。

如何修正可序列化隔離錯誤

錯誤：1023 詳細資訊：Redshift 中的資料表上發生可序列化隔離違規

當 Amazon Redshift 偵測到可序列化隔離錯誤時，您會看到錯誤訊息，如下所示。

```
ERROR:1023 DETAIL: Serializable isolation violation on table in Redshift
```

若要解決可序列化隔離錯誤，您可以嘗試以下方法：

- 重試已取消的交易。

  Amazon Redshift 偵測到並行工作負載不可序列化。其表明應用程序邏輯中有差距，此問題通常可以透過重試遇到錯誤的交易來解決。如果問題仍然存在，請嘗試使用其他方法。

- 將任何無須位於同一不可分割交易中的操作，移到交易之外。

  當兩個交易中的個別操作以可能影響另一個交易結果的方式交叉參考時，適用此方法。例如，以下兩個工作階段各自啟動一個交易。

```
Session1_Redshift=# begin;
```

```
Session2_Redshift=# begin;
```

  每個交易中 SELECT 陳述式的結果，可能會受到另一個交易中 INSERT 陳述式的影響。換句話說，假設您以任何順序，依序執行以下陳述式。在每種情況下，結果都是其中一個 SELECT 陳述式會比同時執行全部交易多傳回一行。依序執行的操作中，沒有任何執行順序可以產生與並行執行相同的結果。因此，最後一個執行的操作會導致可序列化隔離錯誤。

```
Session1_Redshift=# select * from tab1;
Session1_Redshift=# insert into tab2 values (1);
```

```
Session2_Redshift=# insert into tab1 values (1);
Session2_Redshift=# select * from tab2;
```

  在許多情況下，SELECT 陳述式的結果並不重要。換句話說，交易中操作的不可分割性並不重要。在這些情況下，將 SELECT 陳述式移到交易之外，如以下範例所示。

```
Session1_Redshift=# begin;
Session1_Redshift=# insert into tab1 values (1)
Session1_Redshift=# end;
Session1_Redshift=# select * from tab2;
```

```
Session2_Redshift # select * from tab1;
Session2_Redshift=# begin;
Session2_Redshift=# insert into tab2 values (1)
Session2_Redshift=# end;
```

  在這些範例中，交易中沒有交叉參考。兩個 INSERT 陳述式不會互相影響。在這些範例中，至少有一個順序，可讓交易依序執行並產生與並行執行相同的結果。這表示交易是可序列化的。

- 請透過鎖定每個工作階段中的所有資料表來強制序列化。

LOCK 命令能封鎖可能導致可序列化隔離錯誤的操作。使用 LOCK 命令時，請務必執行以下操作：

- 鎖定受交易影響的所有資料表，包括受交易內部唯讀 SELECT 陳述式影響的資料表。
- 無論操作的執行順序為何，均以相同的順序鎖定資料表。
- 在交易開頭，執行任何操作之前，先鎖定所有資料表。

- 對並行交易使用快照隔離

  使用 ALTER DATABASE 命令搭配快照隔離。若要深入了解 ALTER DATABASE 的 SNAPSHOT 參數，請參閱 參數。

錯誤：1018 詳細資訊：關聯不存在

當您在不同的工作階段中執行並行 Amazon Redshift 操作時，您會看到類似如下所示的錯誤訊息。

```
ERROR: 1018 DETAIL: Relation does not exist.
```

Amazon Redshift 中的交易會遵循快照隔離。交易開始後，Amazon Redshift 會擷取資料庫的快照。對於交易的整個生命週期，交易會根據快照中反映的資料庫狀態操作。如果交易從快照集中不存在的資料表進行讀取，則會擲回先前顯示的 1018 錯誤訊息。即使另一個並行交易在交易擷取快照集後建立資料表，交易也無法從新建立的資料表中進行讀取。

為了解決此序列化隔離錯誤，您可以嘗試將交易的開始移至您知道該資料表存在的位置。

如果該資料表是由另一個交易建立，則這一個位置點應至少在已遞交該交易之後。此外，請確定沒有遞交任何可能捨棄資料表的並行交易。

```
session1 = # BEGIN;
session1 = # DROP TABLE A;
session1 = # COMMIT;
```

```
session2 = # BEGIN;
```

```
session3 = # BEGIN;
session3 = # CREATE TABLE A (id INT);
session3 = # COMMIT;
```

```
session2 = # SELECT * FROM A;
```

session2 執行為讀取操作的最後一個操作會導致可序列化隔離錯誤。當 session2 擷取快照集，而資料表已被遞交的 session1 捨棄時，就會發生這個錯誤。換句話說，即使並行的 session3 已經建立資料表，但 session2 仍看不到該資料表，因為其不在快照中。

若要解決此錯誤，您可以按照下方式重新排序工作階段。

```
session1 = # BEGIN;
session1 = # DROP TABLE A;
session1 = # COMMIT;
```

```
session3 = # BEGIN;
session3 = # CREATE TABLE A (id INT);
session3 = # COMMIT;
```

```
session2 = # BEGIN;
session2 = # SELECT * FROM A;
```

現在，當 session2 擷取其快照時，session3 已經遞交，且該資料表位於資料庫中。Session2 可以從資料表中讀取而不會出現任何錯誤。

寫入和讀/寫操作

透過決定何時與如何執行不同的類型的命令，您可以管理並行寫入操作的特定行為。以下是與此討論相關的命令：

- COPY 命令，執行載入 (初始或遞增)
- INSERT 命令，一次附加一或多個資料列
- UPDATE 命令，修改現有資料列
- DELETE 命令，移除資料列

COPY 和 INSERT 操作為單純寫入的操作，但 DELETE 和 UPDATE 操作為讀/寫操作。(若要讓資料列可供刪除或更新，必須先加以讀取。) 並行寫入操作的結果取決於並行執行的特定命令。對相同資料表的 COPY 和 INSERT 操作會保留在等候狀態，直到鎖定釋出為止，然後會照常進行。

UPDATE 和 DELETE 操作有不同的行為，因為它們仰賴於執行任何寫入之前的初始資料表讀取。由於並行交易彼此看不見對方，UPDATE 和 DELETE 必須從上次認可讀取資料的快照。第一個 UPDATE 或 DELETE 釋出其鎖定時，第二個 UPDATE 或 DELETE 必須判斷要使用的資料是否可能過時。它將不會過時，因為第二個交易不會取得其資料的快照，直到第一個交易已釋出其鎖定為止。

並行寫入交易可能的死鎖情況

每當交易牽涉到一個以上資料表的更新，當它們都嘗試寫入相同的資料表集時，並行執行的交易有可能變得死鎖。交易會在認可或復原時一次釋出其所有資料表鎖定；它不會一次撤回一個鎖定。

例如，假設交易 T1 和 T2 在大約相同的時間開始。如果 T1 開始寫入資料表和 T2 開始寫入資料表 B，這兩個交易都可以繼續而不衝突；不過，如果 T1 完成寫入資料表並且必須開始寫入資料表 B，它將無法繼續，因為 T2 仍在 B 上保有鎖定。相反地，如果 T2 完成寫入資料表 B 並且必須開始寫入資料表 A，它將無法繼續，因為 T1 仍在 A 上保有鎖定。因為這兩個交易都可釋出其鎖定，直到其所有寫入操作都認可，因此沒有任一交易可以繼續。

為了避免這類的死鎖，您必須謹慎排程並行寫入操作。例如，您應該一律以相同的順序更新交易中的資料表，同時，如果指定鎖定，在執行任何 DML 操作之前，以相同的順序鎖定資料表。

並行寫入範例

下列虛擬程式碼範例示範交易如何繼續或在並行執行時等候。

對相同的資料表進行並行的 COPY 操作

交易 1 會將資料列複製到 LISTING 資料表：

```
begin;
copy listing from ...;
end;
```

交易 2 會在個別的工作階段中並行開始，並嘗試複製更多資料列至 LISTING 資料表。交易 2 必須等候交易 1 釋出 LISTING 資料表上的寫入鎖定，然後才能繼續。

```
begin;
[waits]
copy listing from ;
end;
```

如果一或兩個交易包含了 INSERT 命令而非 COPY 命令，可能發生相同的行為。

從相同的資料表進行並行的 DELETE 操作

交易 1 會從資料表刪除資料列：

```
begin;
```

```
delete from listing where ...;
end;
```

交易 2 會並行開始，並嘗試從相同的資料表刪除資料列。它會成功，因為它會在嘗試刪除資料列之前等候交易 1 完成。

```
begin
[waits]
delete from listing where ;
end;
```

如果一或兩個交易包含了對相同資料表的 UPDATE 命令而非 DELETE 命令，可能發生相同的行為。

使用讀取和寫入操作混合的並行交易

在此範例中，交易 1 會在認可之前，從 USERS 資料表刪除資料列，重新載入資料表，執行 COUNT(*) 查詢，然後 ANALYZE：

```
begin;
delete one row from USERS table;
copy ;
select count(*) from users;
analyze ;
end;
```

同時，交易 2 會開始。此交易會嘗試複製額外的資料列至 USERS 資料表，分析資料表，然後執行與第一個交易相同的 COUNT(*) 查詢：

```
begin;
[waits]
copy users from ...;
select count(*) from users;
analyze;
end;
```

第二個交易將會成功，因為它必須等候第一個完成。其 COUNT 查詢將根據它完成的載入傳回計數。

教學課程：從 Amazon S3 載入資料

本教學將從頭到尾引導您完成從 Amazon S3 儲存貯體中的資料檔案將資料載入 Amazon Redshift 資料庫資料表的過程。

在此教學課程中，您將執行下列操作：

- 下載使用逗點分隔值 (CSV)、字元分隔或固定寬度格式的資料檔案。
- 建立 Amazon S3 儲存貯體，然後上傳資料檔案至該儲存貯體。
- 啟動 Amazon Redshift 叢集並建立資料庫資料表。
- 使用 COPY 命令從 Amazon S3 上的資料檔案載入資料表。
- 對載入錯誤進行故障診斷，並修改 COPY 命令來更正錯誤。

預估時間：60 分鐘

估計費用：叢集每小時 $1.00

必要條件

您需要以下的事前準備：

- 用於啟動 Amazon Redshift 集群並在 Amazon S3 中創建存儲桶的 AWS 帳戶。
- 用於從 Amazon S3 載入測試資料的 AWS 登入資料 (IAM 角色)。如果您需要新的 IAM 角色，請前往建立 IAM 角色。
- SQL 用戶端，例如 Amazon Redshift 主控台查詢編輯器。

本教學課程設計為可獨立進行。除了本教學課程之外，我們也建議您完成下列教學課程，以便更全面地了解如何設計和使用 Amazon Redshift 資料庫：

- Amazon Redshift 入門指南會逐步引導您完成建立 Amazon Redshift 叢集和載入範例資料的程序。

概觀

您可以使用 INSERT 命令或 COPY 命令將資料新增至您的 Amazon Redshift 資料表。以 Amazon Redshift 資料倉儲的規模和速度，COPY 命令比 INSERT 命令快許多倍且更有效率。

COPY 命令會使用 Amazon Redshift 大量平行處理 (MPP) 架構，從多個資料來源平行讀取和載入資料。您可以從 Amazon S3、Amazon EMR 或任何可透過 Secure Shell (SSH) 連線存取遠端主機上的資料檔案進行載入。或者，您可以直接從 Amazon DynamoDB 資料表載入。

在本教學中，您會使用 COPY 命令從 Amazon S3 載入資料。在此運用的許多原則也適合用於從其他資料來源載入。

若要進一步了解 COPY 命令的使用，請參閱以下資源：

- 載入資料的 Amazon Redshift 最佳實務
- 從 Amazon EMR 載入資料
- 從遠端主機載入資料
- 從 Amazon DynamoDB 資料表載入資料

步驟

步驟 1：建立叢集

如果您已有想要使用的叢集，則可略過此步驟。

針對本教學中的練習，請使用四個節點的叢集。

建立叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。

   在導覽功能表上，選擇已佈建的叢集儀表板。

   > **Important**
   >
   > 請確定您具備執行叢集操作的必要許可。如需授予必要許可的相關資訊，請參閱授權 Amazon Redshift 存取 AWS服務。

2. 在右上角，選擇您要在其中建立叢集的 AWS 區域。基於本教學的用途，請選擇美國西部 (奧勒岡)。
3. 在導覽選單上，選擇叢集，然後選擇建立叢集。建立叢集頁面隨即出現。
4. 在建立叢集頁面上輸入叢集的參數。為參數選擇您自己的值，但變更下列值時除外：
   - 選擇 **dc2.large** 作為節點類型。
   - 選擇 **4** 作為節點數目。
   - 在叢集許可區段中，從可用 IAM 角色中選擇 IAM 角色。此角色應為您先前建立的角色，且具備 Amazon S3 的存取權限。然後選擇與 IAM 角色建立關聯來將其新增至叢集的已關聯 IAM 角色清單。
5. 選擇建立叢集。

遵循 Amazon Redshift 入門指南中的步驟，從 SQL 用戶端連線至您的叢集，並測試連線。您不需要完成「入門」其餘的建立資料表、上傳資料、嘗試範例查詢步驟。

下一步驟

步驟 2：下載資料檔案

步驟 2：下載資料檔案

在此步驟，您會下載一組範例資料檔案到您的電腦。在下一個步驟，您會將檔案上傳到 Amazon S3 儲存貯體。

下載資料檔案

1. 下載壓縮檔案：LoadingDataSampleFiles.zip。
2. 將檔案解壓縮至您電腦中的資料夾。
3. 確認資料夾中包含下列檔案。

```
customer-fw-manifest
customer-fw.tbl-000
customer-fw.tbl-000.bak
customer-fw.tbl-001
customer-fw.tbl-002
customer-fw.tbl-003
customer-fw.tbl-004
customer-fw.tbl-005
```

```
customer-fw.tbl-006
customer-fw.tbl-007
customer-fw.tbl.log
dwdate-tab.tbl-000
dwdate-tab.tbl-001
dwdate-tab.tbl-002
dwdate-tab.tbl-003
dwdate-tab.tbl-004
dwdate-tab.tbl-005
dwdate-tab.tbl-006
dwdate-tab.tbl-007
part-csv.tbl-000
part-csv.tbl-001
part-csv.tbl-002
part-csv.tbl-003
part-csv.tbl-004
part-csv.tbl-005
part-csv.tbl-006
part-csv.tbl-007
```

下一步驟

步驟 3：上傳檔案至 Amazon S3 儲存貯體

步驟 3：上傳檔案至 Amazon S3 儲存貯體

在此步驟，您要建立 Amazon S3 儲存貯體，並將資料檔案上傳至該儲存貯體。

將檔案上傳至 Amazon S3 儲存貯體

1. 在 Amazon S3 中建立儲存貯體。

   如需有關建立儲存貯體的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的建立儲存貯體。

   a. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。

   b. 選擇建立儲存貯體。

   c. 選擇一個 AWS 區域.

在和叢集相同的區域中建立儲存貯體。如果叢集位於美國西部 (奧勒岡) 區域，請選擇美國西部 (奧勒岡) 區域 (us-west-2)。

d. 在建立儲存貯體對話方塊的儲存貯體名稱中，輸入儲存貯體名稱。

在 Amazon S3 現有的所有儲存貯體名稱中，您選擇的儲存貯體名稱必須是唯一的。其中一種協助確保唯一性的方法是在儲存貯體名稱的前面加上組織名稱。儲存貯體名稱必須符合一些規則。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的儲存貯體限制與局限。

e. 為其餘選項選擇建議的預設值。

f. 選擇建立儲存貯體。

當 Amazon S3 成功建立您的儲存貯體，主控台會在儲存貯體面板中顯示您的空儲存貯體。

2. 建立資料夾。

a. 選擇新儲存貯體的名稱。

b. 選擇建立資料夾按鈕。

c. 將新資料夾命名為 **load**。

> **Note**
>
> 您建立的儲存貯體不在沙盒中。在此練習中，您會將物件新增至真正的儲存貯體。您需要為您在儲存貯體中儲存物件的期間支付一筆名目費用。如需 Amazon S3 定價的相關資訊，請移至 Amazon S3 定價頁面。

3. 將資料檔案上傳至新的 Amazon S3 儲存貯體。

a. 選擇資料夾的名稱。

b. 在上傳精靈中，選擇新增檔案。

遵循 Amazon S3 主控台的指示，上傳您下載並擷取的所有檔案。

c. 選擇上傳。

使用者登入資料

Amazon Redshift COPY 命令必須具有讀取 Amazon S3 儲存貯體中檔案物件的存取權。如果您使用相同的使用者登入資料來建立 Amazon S3 儲存貯體以及執行 Amazon Redshift COPY 命令，COPY 命

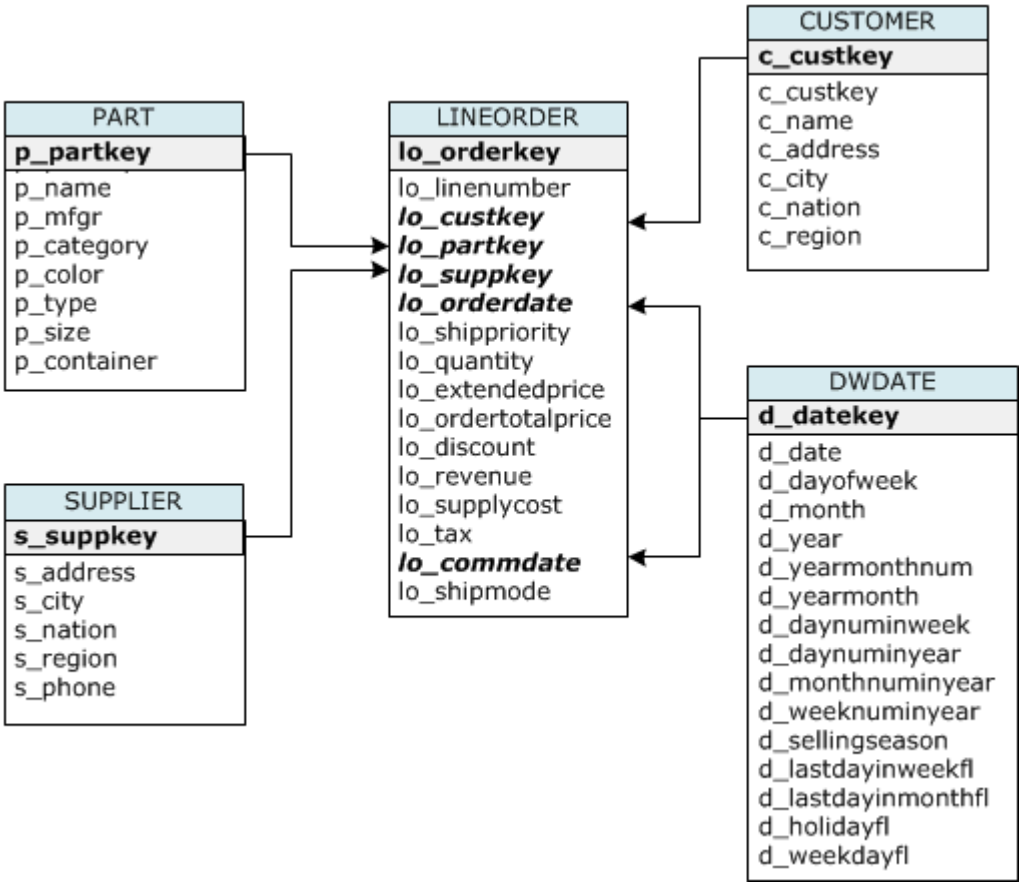
令會具備所有必要的許可。如果您要使用不同的使用者登入資料，可以使用 Amazon S3 存取控制來授予存取權。Amazon Redshift COPY 命令至少需要 ListBucket 和 GetObject 許可才能訪問 Amazon S3 存儲桶中的文件對象。如需對 Amazon S3 資源的存取控制相關資訊，請參閱管理對 Amazon S3 資源的存取許可。

下一步驟

步驟 4：建立範例資料表

步驟 4：建立範例資料表

在本教學課程中，您會使用五個根據星狀結構描述基準化分析 (SSB) 結構描述的資料表。下圖顯示 SSB 資料模型。



SSB 資料表可能已存在於目前的資料庫中。如果目前資料庫中已有 SSB 資料表，您必須先卸除資料表，將其從資料庫中移除，然後在下一個步驟使用 CREATE TABLE 命令建立。本教學課程中使用的資料表可能會有和現有資料表不同的屬性。

建立範例資料表

1. 若要捨棄 SSB 資料表，請在 SQL 用戶端中執行下列命令。

```
drop table part cascade;
drop table supplier;
drop table customer;
drop table dwdate;
drop table lineorder;
```

2. 在 SQL 用戶端中執行下列 CREATE TABLE 命令。

```
CREATE TABLE part
(
  p_partkey     INTEGER NOT NULL,
  p_name        VARCHAR(22) NOT NULL,
  p_mfgr        VARCHAR(6),
  p_category    VARCHAR(7) NOT NULL,
  p_brand1      VARCHAR(9) NOT NULL,
  p_color       VARCHAR(11) NOT NULL,
  p_type        VARCHAR(25) NOT NULL,
  p_size        INTEGER NOT NULL,
  p_container   VARCHAR(10) NOT NULL
);

CREATE TABLE supplier
(
  s_suppkey   INTEGER NOT NULL,
  s_name      VARCHAR(25) NOT NULL,
  s_address   VARCHAR(25) NOT NULL,
  s_city      VARCHAR(10) NOT NULL,
  s_nation    VARCHAR(15) NOT NULL,
  s_region    VARCHAR(12) NOT NULL,
  s_phone     VARCHAR(15) NOT NULL
);

CREATE TABLE customer
(
  c_custkey      INTEGER NOT NULL,
  c_name         VARCHAR(25) NOT NULL,
  c_address      VARCHAR(25) NOT NULL,
  c_city         VARCHAR(10) NOT NULL,
  c_nation       VARCHAR(15) NOT NULL,
```

```
  c_region        VARCHAR(12) NOT NULL,
  c_phone         VARCHAR(15) NOT NULL,
  c_mktsegment    VARCHAR(10) NOT NULL
);

CREATE TABLE dwdate
(
  d_datekey            INTEGER NOT NULL,
  d_date               VARCHAR(19) NOT NULL,
  d_dayofweek          VARCHAR(10) NOT NULL,
  d_month              VARCHAR(10) NOT NULL,
  d_year               INTEGER NOT NULL,
  d_yearmonthnum       INTEGER NOT NULL,
  d_yearmonth          VARCHAR(8) NOT NULL,
  d_daynuminweek       INTEGER NOT NULL,
  d_daynuminmonth      INTEGER NOT NULL,
  d_daynuminyear       INTEGER NOT NULL,
  d_monthnuminyear     INTEGER NOT NULL,
  d_weeknuminyear      INTEGER NOT NULL,
  d_sellingseason      VARCHAR(13) NOT NULL,
  d_lastdayinweekfl    VARCHAR(1) NOT NULL,
  d_lastdayinmonthfl   VARCHAR(1) NOT NULL,
  d_holidayfl          VARCHAR(1) NOT NULL,
  d_weekdayfl          VARCHAR(1) NOT NULL
);
CREATE TABLE lineorder
(
  lo_orderkey          INTEGER NOT NULL,
  lo_linenumber        INTEGER NOT NULL,
  lo_custkey           INTEGER NOT NULL,
  lo_partkey           INTEGER NOT NULL,
  lo_suppkey           INTEGER NOT NULL,
  lo_orderdate         INTEGER NOT NULL,
  lo_orderpriority     VARCHAR(15) NOT NULL,
  lo_shippriority      VARCHAR(1) NOT NULL,
  lo_quantity          INTEGER NOT NULL,
  lo_extendedprice     INTEGER NOT NULL,
  lo_ordertotalprice   INTEGER NOT NULL,
  lo_discount          INTEGER NOT NULL,
  lo_revenue           INTEGER NOT NULL,
  lo_supplycost        INTEGER NOT NULL,
  lo_tax               INTEGER NOT NULL,
  lo_commitdate        INTEGER NOT NULL,
  lo_shipmode          VARCHAR(10) NOT NULL
```

```
);
```

下一步驟

步驟 5：執行 COPY 命令

步驟 5：執行 COPY 命令

您會執行 COPY 命令載入 SSB 結構描述中的每個資料表。在 COPY 命令範例中，將示範使用 COPY 命令的幾個選項從不同檔案格式載入資料，以及進行載入錯誤的故障診斷。

主題

- COPY 命令語法
- 載入 SSB 資料表

COPY 命令語法

COPY 命令基本語法如下。

```
COPY table_name [ column_list ] FROM data_source CREDENTIALS access_credentials
 [options]
```

執行 COPY 命令需提供下列值。

資料表名稱

COPY 命令的目標資料表。此資料表必須已存在於資料庫中。此資料表可以是暫時性或持久性。COPY 命令會將新的輸入資料附加到資料表中任何現有的資料列。

資料欄清單

根據預設，COPY 會按照順序從來源資料將欄位載入資料表的資料欄。您可以選擇指定資料欄清單 (以逗號分隔資料欄名稱的清單)，以便將資料欄位映射到特定的資料欄。您在本教學中不會使用資料行清單。如需詳細資訊，請參閱 COPY 命令參考資料中的Column List。

資料來源

您可以使用 COPY 命令從 Amazon S3 儲存貯體、Amazon EMR 叢集、透過 SSH 連線的遠端主機、或 Amazon DynamoDB 資料表載入資料。在本教學中，您會從 Amazon S3 儲存貯體中的資料檔案載

入資料。從 Amazon S3 進行載入時，您必須提供儲存貯體的名稱和資料檔案的位置。若要執行此作業，請提供資料檔案的物件路徑，或是明確列出每個資料檔案及其位置的資訊清單檔案位置。

- 索引鍵字首

  使用物件索引鍵可以唯一識別儲存在 Amazon S3 中的物件。物件索引鍵包含儲存貯體名稱、資料夾名稱、物件名稱 (如果有)。物件索引鍵字首是指具有相同字首的多個物件。物件路徑就是一種索引鍵字首，COPY 命令用來用來載入具有該索引鍵字首的所有物件。例如，索引鍵字首 `custdata.txt` 可以是指單一檔案或數個檔案，包括 `custdata.txt.001`、`custdata.txt.002` 等。
- 清單檔案

  在某些案例中，您可能需要載入具有不同字首的檔案，例如從多個儲存貯體或資料夾進行載入。在其他案例中，您可能需要排除具有特定字首的檔案。在這些案例中，您可以使用資訊清單檔案。資訊清單檔案會明確列出每個載入檔案及其唯一物件索引鍵。稍後在本教學中，您會使用資訊清單檔案載入 PART 資料表。

登入資料

若要存取包含要載入之資料的 AWS 資源，您必須為具有足夠權限的使用者提供 AWS 存取認證。這些登入資料包括 IAM 角色 Amazon Resource Name (ARN)。若要從 Amazon S3 載入資料，登入資料必須包含 ListBucket 和 GetObject許可。如果您的資料已加密，則需要其他登入資料。如需詳細資訊，請參閱 COPY 命令參考資料中的授權參數。如需管理存取權的相關資訊，請前往管理對 Amazon S3 資源的存取許可。

選項

您可以指定數個參數搭配 COPY 命令，藉此指定檔案格式、管理資料格式、管理錯誤、控制其他功能。在本教學中，您會使用下列 COPY 命令選項和功能：

- 索引鍵字首

  如需如何透過指定金鑰前綴從多個檔案載入的相關資訊，請參閱使用 NULL AS 載入 PART 資料表。
- CSV format (CSV 格式)

  如需如何載入 CSV 格式資料的相關資訊，請參閱使用 NULL AS 載入 PART 資料表。
- NULL AS

  如需如何使用 NULL AS 選項載入 PART 的相關資訊，請參閱使用 NULL AS 載入 PART 資料表。
- 字元分隔的格式

如需如何使用 DELIMITER 選項的相關資訊，請參閱使用 REGION 載入 SUPPLIER 資料表。

- REGION

  如需如何使用 REGION 選項的相關資訊，請參閱使用 REGION 載入 SUPPLIER 資料表。

- 固定格式寬度

  如需如何從固定寬度資料載入 CUSTOMER 資料表的相關資訊，請參閱使用 MANIFEST 載入 CUSTOMER 資料表。

- MAXERROR

  如需如何使用 MAXERROR 選項的相關資訊，請參閱使用 MANIFEST 載入 CUSTOMER 資料表。

- ACCEPTINVCHARS

  如需如何使用 ACCEPTINVCHARS 選項的相關資訊，請參閱使用 MANIFEST 載入 CUSTOMER 資料表。

- MANIFEST

  如需如何使用 MANIFEST 選項的相關資訊，請參閱使用 MANIFEST 載入 CUSTOMER 資料表。

- DATEFORMAT

  如需如何使用 DATEFORMAT 選項的相關資訊，請參閱使用 DATEFORMAT 載入 DWDATE 資料表。

- GZIP、LZOP 與 BZIP2

  如需有關如何壓縮檔案的相關資訊，請參閱使用多個檔案載入 LINEORDER 資料表。

- COMPUPDATE

  如需如何使用 COMPUPDATE 選項的相關資訊，請參閱使用多個檔案載入 LINEORDER 資料表。

- 多個檔案

  如需如何載入多個檔案的相關資訊，請參閱使用多個檔案載入 LINEORDER 資料表。

載入 SSB 資料表

您會使用下列 COPY 命令載入 SSB 結構描述中的每個資料表。每個資料表的命令分別示範不同的 COPY 選項以及故障診斷技巧。

若要載入 SSB 資料表，請遵循以下步驟：

1. 取代值區名稱和 AWS 認證
2. 使用 NULL AS 載入 PART 資料表
3. 使用 REGION 載入 SUPPLIER 資料表
4. 使用 MANIFEST 載入 CUSTOMER 資料表
5. 使用 DATEFORMAT 載入 DWDATE 資料表
6. 使用多個檔案載入 LINEORDER 資料表

取代值區名稱和 AWS 認證

本教學課程中的 COPY 命令為以下格式。

```
copy table from 's3://<your-bucket-name>/load/key_prefix'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
options;
```

針對所有 COPY 命令，執行下列操作：

1. 將 *<your-bucket-name>* 取代為與您的叢集位於同一區域的儲存貯體名稱。

   本步驟假設儲存貯體與叢集位於同一區域。或者，您可以使用 REGION 選項搭配 COPY 命令來指定區域。

2. <aws-account-id><role-name>以您自己 *AWS ##* 和 IAM 角色取代和。以單引號括住的登入資料字串區段不可包含任何空格或分行符號。請注意，ARN 的格式可能與範例略有不同。最好從 IAM 主控台複製角色的 ARN，以確保在執行 COPY 命令時正確無誤。

使用 NULL AS 載入 PART 資料表

在此步驟中，您將使用 CSV 和 NULL AS 選項載入 PART 資料表。

COPY 命令可以從多個檔案平行載入資料，這比從單一檔案載入更快速。為了示範這個原則，本教學課程將每個資料表中的資料分成八個檔案，即使檔案很小。在稍後的步驟中，您會比較從單一檔案與從多個檔案載入之間的時間差異。如需詳細資訊，請參閱 載入資料檔案。

索引鍵字首

您可以透過為數個檔案指定索引鍵字首，或在資訊清單檔案中明確列出檔案，來從多個檔案載入資料。在此步驟中，您會使用金鑰前綴。在稍後的步驟中，您會使用資訊清單檔案。's3://mybucket/load/part-csv.tbl' 索引鍵字首會載入 load 資料夾中的下列這些檔案。

```
part-csv.tbl-000
part-csv.tbl-001
part-csv.tbl-002
part-csv.tbl-003
part-csv.tbl-004
part-csv.tbl-005
part-csv.tbl-006
part-csv.tbl-007
```

CSV format (CSV 格式)

CSV 意指以逗號分隔值，是用於匯入和匯出試算表資料的常見格式。CSV 比逗號分隔的格式更有彈性，因為它可讓您在欄位中包含引用字串。從 CSV 格式 COPY 的預設引號字元是雙引號 (")，但您可以使用 QUOTE AS 選項指定其他引號字元。在欄位內使用引號字元時，請多加一個引號字元來逸出此字元。

以下摘錄自 PART 資料表的 CSV 格式資料檔案，顯示以雙引號括起來的字串 ("LARGE ANODIZED BRASS")。其也顯示了引用字串中以兩個雙引號括起來的字串 ("MEDIUM ""BURNISHED"" TIN")。

```
15,dark sky,MFGR#3,MFGR#47,MFGR#3438,indigo,"LARGE ANODIZED BRASS",45,LG CASE
22,floral beige,MFGR#4,MFGR#44,MFGR#4421,medium,"PROMO, POLISHED BRASS",19,LG DRUM
23,bisque slate,MFGR#4,MFGR#41,MFGR#4137,firebrick,"MEDIUM ""BURNISHED"" TIN",42,JUMBO
 JAR
```

PART 資料表的資料包含會導致 COPY 失敗的字元。在此練習中，您會對錯誤進行故障診斷，並修正錯誤。

若要載入 CSV 格式的資料，請在 COPY 命令中加入 csv。執行下列命令載入 PART 資料表。

```
copy part from 's3://<your-bucket-name>/load/part-csv.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
csv;
```

您可能會看到類似下列的錯誤訊息。

```
An error occurred when executing the SQL command:
```

```
copy part from 's3://mybucket/load/part-csv.tbl'
credentials' ...

ERROR: Load into table 'part' failed.  Check 'stl_load_errors' system table for
 details. [SQL State=XX000]

Execution time: 1.46s

1 statement(s) failed.
1 statement(s) failed.
```

若要取得有關錯誤的詳細資訊，請查詢 STL_LOAD_ERRORS 資料表。下列查詢使用 SUBSTRING 函數縮短資料欄以利閱讀，並使用 LIMIT 10 減少傳回的資料列數。您可以調整 `substring(filename,22,25)` 中的值，以符合您的儲存貯體名稱長度。

```
select query, substring(filename,22,25) as filename,line_number as line,
substring(colname,0,12) as column, type, position as pos, substring(raw_line,0,30) as
 line_text,
substring(raw_field_value,0,15) as field_text,
substring(err_reason,0,45) as reason
from stl_load_errors
order by query desc
limit 10;
```

```
 query  |    filename      | line |  column   |    type    | pos |
--------+------------------------+-----------+------------+------------+-----+----
 333765 | part-csv.tbl-000 |    1 |           |            |   0 |

 line_text        | field_text |                    reason
------------------+------------+-----------------------------------------------
 15,NUL next,     |            | Missing newline: Unexpected character 0x2c f
```

NULL AS

`part-csv.tbl` 資料檔案使用 NUL 結束字元 (`\x000` 或 `\x0`) 來指出 NULL 值。

> **Note**
>
> 雖然看起來很像，但 NUL 和 NULL 大不相同。NUL 是有 `x000` 字碼指標的 UTF-8 字元，通常用於表示記錄結束 (EOR)。NULL 是用來代表缺少某個值的 SQL 值。

根據預設，COPY 會將 NUL 結束字元視為 EOR 字元並終止記錄，而這常會造成未預期的結果或錯誤。指出文字資料中的 NULL 沒有任何單一的標準方法。因此，NULL AS COPY 命令選項可讓您指定在載入資料表時，要用哪個字元替換 NULL。在此範例中，要讓 COPY 將 NUL 結束字元當作 NULL 值。

> **Note**
>
> 必須將要接收 NULL 值的資料表欄設為 nullable。也就是說，在 CREATE TABLE 規格中，它一定不能包含 NOT NULL 限制。

若要使用 NULL AS 選項載入 PART，請執行以下 COPY 命令。

```
copy part from 's3://<your-bucket-name>/load/part-csv.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
csv
null as '\000';
```

若要確認 COPY 已載入 NULL 值，執行下列命令可以僅選取包含 NULL 的資料列。

```
select p_partkey, p_name, p_mfgr, p_category from part where p_mfgr is null;
```

```
 p_partkey |  p_name  | p_mfgr | p_category
-----------+----------+--------+------------
        15 | NUL next |        | MFGR#47
        81 | NUL next |        | MFGR#23
       133 | NUL next |        | MFGR#44
(2 rows)
```

使用 REGION 載入 SUPPLIER 資料表

在此步驟中，您會使用 DELIMITER 和 REGION 選項載入 SUPPLIER 資料表。

> **Note**
>
> 用於載入 SUPPORDER 表格的檔案會在 AWS 範例值區中提供。您不需要在這個步驟上傳檔案。

字元分隔的格式

字元分隔檔案中的欄位是以特殊字元隔開，例如縱線字元 (|)、逗號 (,) 或 Tab 字元 (\t)。字元分隔檔案可以使用任可單一 ASCII 字元做為分隔符號，包括其中一個非列印 ASCII 字元。請使用 DELIMITER 選項指定分隔符號。預設分隔符號是縱線字元 (|)。

下列摘錄自 SUPPLIER 資料表的資料使用縱線分隔格式。

```
1|1|257368|465569|41365|19950218|2-HIGH|0|17|2608718|9783671|4|2504369|92072|2|
19950331|TRUCK
1|2|257368|201928|8146|19950218|2-HIGH|0|36|6587676|9783671|9|5994785|109794|6|
19950416|MAIL
```

REGION

只要有可能，您應該在 Amazon Redshift 叢集所在的相同 AWS 區域中找到負載資料。如果您的資料和叢集位於相同區域中，可以降低延遲，並且避免跨區域傳輸資料的成本。如需更多資訊，請參閱載入資料的 Amazon Redshift 最佳實務

如果您必須從其他 AWS 區域載入資料，請使用 REGION 選項來指定載入資料所在的 AWS 區域。如果指定區域，則所有載入資料 (包括資訊清單檔案) 都必須位於指定的區域。如需詳細資訊，請參閱 REGION。

如果您的叢集位於美國東部 (維吉尼亞北部)，執行以下命令從位於美國西部 (奧勒岡) 區域的 Amazon S3 儲存貯體上的縱線分隔資料載入 SUPPLIER 資料表。在這個範例中，請勿變更儲存貯體名稱。

```
copy supplier from 's3://awssampledbuswest2/ssbgz/supplier.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
delimiter '|'
gzip
region 'us-west-2';
```

如果您的叢集不是位於美國東部 (維吉尼亞北部)，請執行以下命令，從位於美國東部 (維吉尼亞北部) 區域的 Amazon S3 儲存貯體上的縱線分隔資料載入 SUPPLIER 資料表。在這個範例中，請勿變更儲存貯體名稱。

```
copy supplier from 's3://awssampledb/ssbgz/supplier.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
delimiter '|'
gzip
region 'us-east-1';
```

使用 MANIFEST 載入 CUSTOMER 資料表

在此步驟中，您會使用 FIXEDWIDTH、MAXERROR、ACCEPTINVCHARS 和 MANIFEST 選項載入 CUSTOMER 資料表。

此練習的範例資料包含 COPY 嘗試載入時會造成錯誤的字元。您會使用 MAXERRORS 選項和 STL_LOAD_ERRORS 系統資料表，針對載入錯誤進行故障診斷，然後使用 ACCEPTINVCHARS 和 MANIFEST 選項消除錯誤。

固定寬度格式

固定寬度格式將每個欄位定義為固定數量的字元，而不是使用分隔符號分隔欄位。下列摘錄自 CUSTOMER 資料表的資料使用固定寬度格式。

```
1   Customer#000000001   IVhzIApeRb            MOROCCO  0MOROCCO  AFRICA      25-705
2   Customer#000000002   XSTf4,NCwDVaWNe6tE    JORDAN   6JORDAN   MIDDLE EAST 23-453
3   Customer#000000003   MG9kdTD               ARGENTINA5ARGENTINAAMERICA     11-783
```

標籤/寬度配對的順序必須完全符合資料表欄的順序。如需詳細資訊，請參閱 FIXEDWIDTH。

CUSTOMER 資料表固定寬度的規格字串如下。

```
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
c_region :12, c_phone:15,c_mktsegment:10'
```

若要從固定寬度資料載入 CUSTOMER 資料表，執行下列命令。

```
copy customer
from 's3://<your-bucket-name>/load/customer-fw.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
 c_region :12, c_phone:15,c_mktsegment:10';
```

您應該會看到類似下列的錯誤訊息。

```
An error occurred when executing the SQL command:
copy customer
from 's3://mybucket/load/customer-fw.tbl'
credentials'...
```

```
ERROR: Load into table 'customer' failed.  Check 'stl_load_errors' system table for
 details. [SQL State=XX000]

Execution time: 2.95s

1 statement(s) failed.
```

MAXERROR

根據預設，COPY 第一次遇到錯誤時，命令就會失敗並傳回錯誤訊息。若要在測試期間節省時間，您可以使用 MAXERROR 選項來指示 COPY 在失敗前可略過指定的錯誤數量。因為我們預料到第一次測試載入 CUSTOMER 資料表資料時會發生錯誤，請在 COPY 命令中加入 `maxerror 10`。

若要使用 FIXEDWIDTH 和 MAXERROR 選項進行測試，執行下列命令。

```
copy customer
from 's3://<your-bucket-name>/load/customer-fw.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
 c_region :12, c_phone:15,c_mktsegment:10'
maxerror 10;
```

這次，您不會看到錯誤訊息，而是得到類似下列的警示訊息。

```
Warnings:
Load into table 'customer' completed, 112497 record(s) loaded successfully.
Load into table 'customer' completed, 7 record(s) could not be loaded.  Check
 'stl_load_errors' system table for details.
```

警示表示 COPY 遇到 7 個錯誤。若要查看錯誤，請查詢 STL_LOAD_ERRORS 資料表，如以下範例所示。

```
select query, substring(filename,22,25) as filename,line_number as line,
substring(colname,0,12) as column, type, position as pos, substring(raw_line,0,30) as
 line_text,
substring(raw_field_value,0,15) as field_text,
substring(err_reason,0,45) as error_reason
from stl_load_errors
order by query desc, filename
limit 7;
```

查詢 STL_LOAD_ERRORS 的結果應如下所示。

```
 query  |         filename          | line |  column   |    type    | pos |
 line_text           | field_text |              error_reason
--------+---------------------------+------+-----------
+------------+-----+-------------------------------+------------
+--------------------------------------------
 334489 | customer-fw.tbl.log       |    2 | c_custkey | int4       |  -1 | customer-
fw.tbl               | customer-f | Invalid digit, Value 'c', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    6 | c_custkey | int4       |  -1 | Complete
                  | Complete   | Invalid digit, Value 'C', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    3 | c_custkey | int4       |  -1 | #Total rows
                  | #Total row | Invalid digit, Value '#', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    5 | c_custkey | int4       |  -1 | #Status
                  | #Status    | Invalid digit, Value '#', Pos 0, Type: Integ
 334489 | customer-fw.tbl.log       |    1 | c_custkey | int4       |  -1 | #Load file
                  | #Load file | Invalid digit, Value '#', Pos 0, Type: Integ
 334489 | customer-fw.tbl000        |    1 | c_address | varchar    |  34 | 1
 Customer#000000001  | .Mayag.ezR | String contains invalid or unsupported UTF8
 334489 | customer-fw.tbl000        |    1 | c_address | varchar    |  34 | 1
 Customer#000000001  | .Mayag.ezR | String contains invalid or unsupported UTF8
(7 rows)
```

檢查結果時，您可以看到 error_reasons 資料欄中有兩則訊息：

- `Invalid digit, Value '#', Pos 0, Type: Integ`

  這些錯誤是由 customer-fw.tbl.log 檔案引起。問題在於，它是日誌檔案不是資料檔案，不應載入它。您可以使用資訊清單檔案來避免載入錯誤的檔案。

- `String contains invalid or unsupported UTF8`

  VARCHAR 資料類型支援最多 3 個位元組的 UTF-8 多位元組字元。如果載入資料包含不支援或無效的字元，您可以使用 ACCEPTINVCHARS 選項用指定的替代字元取代每個無效字元。

另一個載入的問題比較難偵測 — 載入產生非預期的結果。若要調查這個問題，執行下列命令查詢 CUSTOMER 資料表。

```
select c_custkey, c_name, c_address
from customer
```

```
order by c_custkey
limit 10;
```

```
 c_custkey |          c_name           |         c_address
-----------+---------------------------+---------------------------
         2 | Customer#000000002        | XSTf4,NCwDVaWNe6tE
         2 | Customer#000000002        | XSTf4,NCwDVaWNe6tE
         3 | Customer#000000003        | MG9kdTD
         3 | Customer#000000003        | MG9kdTD
         4 | Customer#000000004        | XxVSJsL
         4 | Customer#000000004        | XxVSJsL
         5 | Customer#000000005        | KvpyuHCplrB84WgAi
         5 | Customer#000000005        | KvpyuHCplrB84WgAi
         6 | Customer#000000006        | sKZz0CsnMD7mp4Xd0YrBvx
         6 | Customer#000000006        | sKZz0CsnMD7mp4Xd0YrBvx
(10 rows)
```

這些資料列應該是獨一無二的，但其中有重複。

另一個檢查非預期結果的方法，是去確有已載入的資料列數量。在我們的案例中，應該載入 100000 列資料，但載入訊息指出載入 112497 個記錄。會載入多的資料列，是因為 COPY 載入無關的檔案 `customer-fw.tbl0000.bak`。

在這個練習中，您會使用資訊清單檔案來避免載入錯誤的檔案。

ACCEPTINVCHARS

根據預設，當 COPY 遇到欄位資料類型不支援的字元，會略過該資料列並傳回錯誤。如需無效 UTF-8 字元的相關資訊，請參閱多位元組字元載入錯誤。

您可以使用 MAXERRORS 選項來略過錯誤並繼續載入，然後查詢 STL_LOAD_ERRORS 來找出無效字元，接著修正資料檔案。不過，MAXERRORS 最好用於進行載入問題的故障診斷，不應廣泛使用在生產環境中。

ACCEPTINVCHARS 選項通常是管理無效字元更好的選擇。ACCEPTINVCHARS 選項會指示 COPY 用指定的有效字元取代每個無效字元，並繼續載入操作。您可以指定 NULL 除外的任何 ASCII 字元做為替代字元。預設的替代字元是問號 (?)。COPY 會將多位元組字元取代為相等長度的替代字元。例如，4 個位元組的字元會被取代為 '????'。

COPY 會傳回包含無效 UTF-8 字元的資料列數。其也會將項目新增至每個受影響資料列的 STL_REPLACEMENTS 系統資料表，每個節點分割最多 100 個資料列。也會取代其他無效 UTF-8 字元，但不會記錄那些取代事件。

ACCEPTINVCHARS 僅適用於 VARCHAR 欄。

您會在此步驟中使用替代字元 '^' 新增 ACCEPTINVCHARS。

MANIFEST

當您使用金鑰前綴從 Amazon S3 COPY 時，其中一個風險是您可能會載入不必要的資料表。例如，'s3://mybucket/load/ 資料夾包含 8 個檔案，它們有相同的索引鍵字首 customer-fw.tbl：customer-fw.tbl0000、customer-fw.tbl0001 等。然而，同一資料夾中也包含無關的檔案 customer-fw.tbl.log 和 customer-fw.tbl-0001.bak。

為了確保載入所有正確的檔案，且只有正確的檔案，請使用資訊清單檔案。資訊清單是 JSON 格式的文字檔案，其中明確列出要載入之每個來源檔案的唯一物件索引鍵。檔案物件可以位於不同的資料夾或儲存貯體，但必須在同一個區域中。如需詳細資訊，請參閱 MANIFEST。

以下顯示 customer-fw-manifest 的文字。

```
{
  "entries": [
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-000"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-001"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-002"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-003"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-004"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-005"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-006"},
    {"url":"s3://<your-bucket-name>/load/customer-fw.tbl-007"}
    ]
}
```

使用資訊清單檔案從 CUSTOMER 資料表載入資料

1. 在文字編輯器中開啟 customer-fw-manifest 檔案。
2. 以您的儲存貯體名稱取代 *<your-bucket-name>*。
3. 儲存檔案。
4. 將檔案上傳到儲存貯體上的 load 資料夾。
5. 執行下列 COPY 命令。

```
copy customer from 's3://<your-bucket-name>/load/customer-fw-manifest'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
```

```
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
 c_region :12, c_phone:15,c_mktsegment:10'
maxerror 10
acceptinvchars as '^'
manifest;
```

使用 DATEFORMAT 載入 DWDATE 資料表

在此步驟中，您會使用 DELIMITER 和 DATEFORMAT 選項載入 DWDATE 資料表。

載入 DATE 和 TIMESTAMP 資料欄時，COPY 期望是預設格式：日期為 YYYY-MM-DD，時間戳記為 YYYY-MM-DD HH:MI:SS。如果載入的資料不是使用預設格式，您可以使用 DATEFORMAT 和 TIMEFORMAT 指定格式。

下列的摘錄顯示了 DWDATE 資料表中的日期格式。注意兩個資料欄的日期格式不一致。

```
19920104 1992-01-04          Sunday  January 1992 199201 Jan1992 1 4 4 1...
19920112 January 12, 1992 Monday  January 1992 199201 Jan1992 2 12 12 1...
19920120 January 20, 1992 Tuesday     January 1992 199201 Jan1992 3 20 20 1...
```

DATEFORMAT

您只能指定一個日期格式。如果載入資料包含不一致的格式 (可能在不同資料欄中)，或是載入時格式未知，請使用 DATEFORMAT 與 'auto' 引數。指定 'auto' 時，COPY 會辨識所有有效的日期或時間格式，並將其轉換為預設格式。'auto' 選項可以辨識使用 DATEFORMAT 和 TIMEFORMAT 字串時不支援的多種格式。如需詳細資訊，請參閱 對 DATEFORMAT 和 TIMEFORMAT 使用自動辨識。

若要載入 DWDATE 資料表，執行下列 COPY 命令。

```
copy dwdate from 's3://<your-bucket-name>/load/dwdate-tab.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
delimiter '\t'
dateformat 'auto';
```

使用多個檔案載入 LINEORDER 資料表

此步驟使用 GZIP 和 COMPUPDATE 選項載入 LINEORDER 資料表。

在這個練習中，您會從單一資料檔案載入 LINEORDER 資料表，再從多個檔案再次載入。執行此作業可讓您比較兩個方法的載入時間。

> **Note**
>
> 用於載入 LINEORDER 表格的檔案會在 AWS 範例儲存貯體中提供。您不需要在這個步驟上傳檔案。

GZIP、LZOP 與 BZIP2

您可以使用 gzip、lzop 或 bzip2 壓縮格式來壓縮您的檔案。從解壓縮檔案載入時，COPY 會在載入程序中將檔案解壓縮。將檔案壓縮可以節省儲存空間及縮短載入時間。

COMPUPDATE

當 COPY 載入無壓縮編碼的空資料表時，會分析載入資料並決定最佳的編碼。接著它會將資料表修改為使用這些編碼，再開始載入。這個分析程序需要一點時間，但幾乎在每個資料表都會發生一次。若要節省時報，您可以關閉 COMPUPDATE 來略過這個步驟。為了準確評估 COPY 時間，您會在這個步驟中關閉 COMPUPDATE。

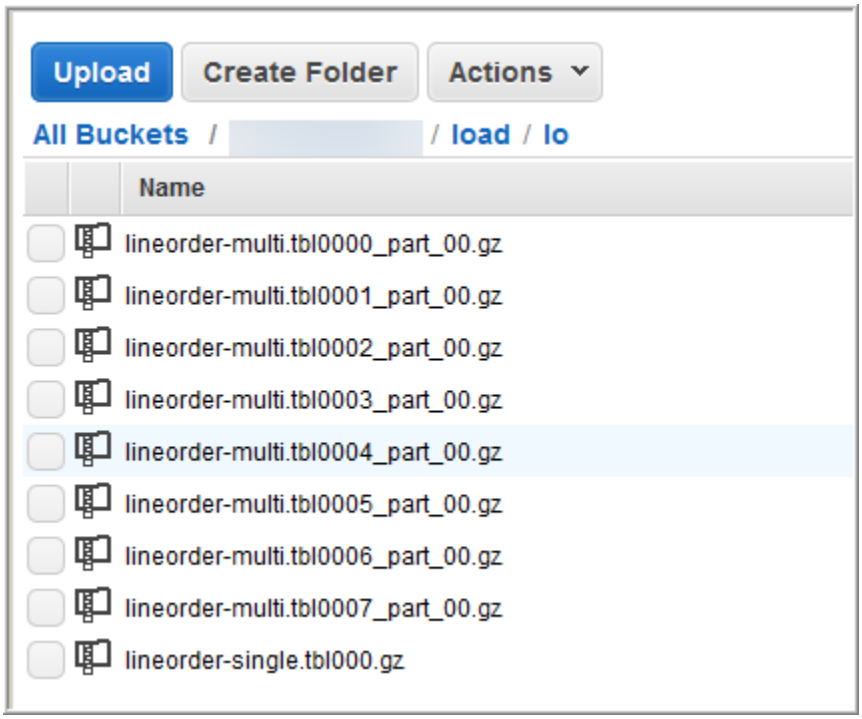
多個檔案

相較於從單一檔案載入資料，COPY 命令從多個檔案平行載入資料更有效率。您可以將您的資料分割為檔案，使得檔案的數量為您叢集中配量數量的倍數。若您執行此作業，Amazon Redshift 會分割工作負載，並在配量中平均分配資料。每一節點的配量數目取決於叢集的節點大小。如需每個節點大小有多少配量的相關資訊，請移至《Amazon Redshift 管理指南》中的關於叢集和節點。

例如，在本教學中使用的 dc2.large 運算節點有兩個配量，所以四個叢集總共有八個配量。先前的步驟將載入的資料分成八個檔案，即使檔案很小。在這個步驟中，您會比較從單一大檔案與從多個檔案載入之間的時間差異。

您用於本教學的檔案約有 1500 萬筆記錄，大小約 1.2 GB。以 Amazon Redshift 的規模來說這些檔案很小，但足以示範從多個檔案載入的效能優點。由於在本教學中這些檔案已夠大，載入這些檔案再將其上傳到 Amazon S3 所需的時間很長。因此，您可以直接從 AWS 範例值區載入檔案。

以下螢幕擷取畫面顯示 LINEORDER 的資料檔案。



評估 COPY 多個檔案的效能

1. 執行下列命令從單一檔案 COPY。請勿變更儲存貯體名稱。

```
copy lineorder from 's3://awssampledb/load/lo/lineorder-single.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
gzip
compupdate off
region 'us-east-1';
```

2. 結果類似以下這樣。請注意執行時間。

```
Warnings:
Load into table 'lineorder' completed, 14996734 record(s) loaded successfully.

0 row(s) affected.
copy executed successfully

Execution time: 51.56s
```

3. 執行下列命令從多個檔案 COPY。請勿變更儲存貯體名稱。

```
copy lineorder from 's3://awssampledb/load/lo/lineorder-multi.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
```

```
gzip
compupdate off
region 'us-east-1';
```

4. 結果類似以下這樣。請注意執行時間。

```
Warnings:
Load into table 'lineorder' completed, 14996734 record(s) loaded successfully.

0 row(s) affected.
copy executed successfully

Execution time: 17.7s
```

5. 比較執行時間。

   在我們的範例中，載入 1500 萬個記錄的時間從 51.56 秒降到 17.7 秒，減少了百分之 65.7。

   這些結果來自於使用具有四個節點的叢集。如果您的叢集有更多節點，節省的時間會加倍。典型的 Amazon Redshift 叢集有數十到數百個節點，差別更巨大。如果您的叢集只有單一節點，執行時間的差異很小。

下一步驟

步驟 6：清空及分析資料庫

步驟 6：清空及分析資料庫

每當您新增、刪除或修改大量資料列時，應先後執行 VACUUM 命令及 ANALYZE 命令。vacuum (清空) 會恢復已刪除資料列的空間，並還原排序。ANALYZE 命令會更新統計中繼資料，後者可讓查詢最佳化工具產生更正確的查詢計畫。如需詳細資訊，請參閱 清空資料表。

如果您以排序索引鍵的順序載入資料，清空很快。在本教學課程中，您已新增大量資料列，但是將它們新增到空資料表。在這種情況下，不需要重新排序，您也沒有刪除任何資料列。COPY 在加載空表後自動更新統計信息，因此您的統計信息應該是 up-to-date。但基於良好的內務處理，您會清空並分析資料庫以完成本教學。

若要清空並分析資料庫，執行下列命令。

```
vacuum;
analyze;
```

下一步驟

步驟 7：清理您的資源

步驟 7：清理您的資源

叢集只要執行就會繼續產生費用。完成本教學課程後，您應按照 Amazon Redshift 入門指南中的步驟 5：撤銷存取權並刪除範例叢集中的步驟，將環境恢復到先前的狀態。

如果您要保留叢集，但又想復原 SSB 資料表所使用的儲存體，請執行下列命令。

```
drop table part;
drop table supplier;
drop table customer;
drop table dwdate;
drop table lineorder;
```

下一頁

Summary

Summary

在本教學課程中，您已將檔案上傳到 Amazon S3，並使用 COPY 命令將資料從該檔案載入 Amazon Redshift 資料表。

您已使用以下格式載入資料：

- 字元分隔
- CSV
- 固定寬度

您已使用 STL_LOAD_ERRORS 系統資料表進行載入錯誤的故障診斷，並使用 REGION、MANIFEST、MAXERROR、ACCEPTINVCHARS、DATEFORMAT、NULL AS 選項解決錯誤。

您已運用下列最佳實務來載入資料：

- 使用 COPY 命令載入資料

- 載入資料檔案
- 使用單一 COPY 命令從多個檔案載入
- 壓縮您的資料檔案
- 在載入前後驗證資料檔案

如需 Amazon Redshift 最佳實務的相關資訊，請參閱下列連結：

- 載入資料的 Amazon Redshift 最佳實務
- 設計資料表的 Amazon Redshift 最佳實務
- 設計查詢的 Amazon Redshift 最佳實務

卸載資料

主題

若要將資料從資料庫表格中卸載至 Amazon S3 儲存貯體中的一組檔案，您可以使用 UNLOAD 命令搭配 SELECT 陳述式。您可以透過分隔的格式或是固定寬度的格式來卸載資料，無論資料載入所使用的格式為何。您也可以指定是否要建立壓縮的 GZIP 檔案。

您也可以使用臨時的安全性登入資料，來限制使用者對 Amazon S3 儲存貯體的存取。

將資料卸載到 Amazon S3

Amazon Redshift 會將 select 陳述式的結果分散在一組檔案中，每個節點分割中一或多個檔案，來簡化資料的平行重新載入。或者，您可以指定 UNLOAD，其應會透過新增 PARALLEL OFF 選項，來依順序將結果寫入至一或多個檔案。您可以透過指定 MAXFILESIZE 參數，來限制 Amazon S3 中的檔案大小。UNLOAD 會自動使用 Amazon S3 伺服器端加密 (SSE-S3) 加密資料檔案。

您可以在 Amazon Redshift 支援的 UNLOAD 命令中使用任何 select 陳述式，除了在外部 select 中使用 LIMIT 子句的 select 以外。例如，您可以使用其中包含特定欄位的 select 陳述式，或使用 where 子句來聯結多個表格。如果您的查詢包含引號 (例如用來括住常值)，您需要在查詢常值中逸出它們 (\')。如需詳細資訊，請參閱 SELECT 命令參考。如需使用 LIMIT 子句的詳細資訊，請參閱 使用須知 以了解 UNLOAD 命令。

例如，下列 UNLOAD 命令可將 VENUE 表格的內容傳送至 Amazon S3 儲存貯體 `s3://mybucket/tickit/unload/`。

```
unload ('select * from venue')
to 's3://mybucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

先前範例建立的檔名包含字首 'venue_'。

```
venue_0000_part_00
```

```
venue_0001_part_00
venue_0002_part_00
venue_0003_part_00
```

依預設，UNLOAD 會根據叢集中的分割數，將資料平行寫入多個檔案。若要將資料寫入單一檔案，請指定 PARALLEL OFF。UNLOAD 會按順序寫入資料，且如果有使用 ORDER BY 字句，則會根據該子句進行絕對排列。資料檔案大小上限為 6.2 GB。如果資料大小大於上限 (每個高達 6.2 GB)，UNLOAD 會額外建立檔案。

以下範例指會將內容 VENUE 寫入單一檔案。僅需要一個檔案，因為檔案大小少於 6.2 GB。

```
unload ('select * from venue')
to 's3://mybucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off;
```

> **Note**
>
> UNLOAD 命令是設計為使用平行處理。我們建議在多數的情況將 PARALLEL 保持為啟用，特別是在使用 COPY 命令將檔案是用來載入表格時。

假設 VENUE 的資料大小總計為 5 GB，下列範例會將 VENUE 的內容寫入至 50 個檔案，每個大小為 100 MB。

```
unload ('select * from venue')
to 's3://mybucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off
maxfilesize 100 mb;
```

如果您在 Amazon S3 路徑中包含字首，UNLOAD 將使用該字首做為檔名。

```
unload ('select * from venue')
to 's3://mybucket/tickit/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

您可以在 UNLOAD 命令指定 MANIFEST 選項，以建立列出卸載檔案的資訊清單檔案。清單檔案是 JSON 格式的文字檔，其中明確列出寫入 Amazon S3 的每個檔案的 URL。

下列範例包含資訊清單選項。

```
unload ('select * from venue')
to 's3://mybucket/tickit/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

下列範例顯示四個卸載檔案的資訊清單。

```
{
  "entries": [
    {"url":"s3://mybucket/tickit/venue_0000_part_00"},
    {"url":"s3://mybucket/tickit/venue_0001_part_00"},
    {"url":"s3://mybucket/tickit/venue_0002_part_00"},
    {"url":"s3://mybucket/tickit/venue_0003_part_00"}
  ]
}
```

資訊清單檔案可用來載入相同的檔案，方法是使用 COPY 搭配 MANIFEST 選項。如需更多詳細資訊，請參閱 使用資訊清單指定資料檔案。

在您完成 UNLOAD 操作後，透過導覽至 UNLOAD 將檔案寫入其中的 Amazon S3 儲存貯體，來確認檔案是否正確卸載。您將會在每個分割看到一或多個含編號的檔案，從編號零開始。如果指定 MANIFEST 選項，則您將會看見結尾為 `manifest` 的檔案。例如：

```
mybucket/tickit/venue_0000_part_00
mybucket/tickit/venue_0001_part_00
mybucket/tickit/venue_0002_part_00
mybucket/tickit/venue_0003_part_00
mybucket/tickit/venue_manifest
```

您可以在 UNLOAD 完成後呼叫 Amazon S3 清單操作，以程式設計方式取得寫入 Amazon S3 的檔案清單。您也可以查詢 STL_UNLOAD_LOG。

下列查詢會傳回已由 UNLOAD 建立之檔案的路徑名稱。PG_LAST_QUERY_ID 函數會傳回最近的查詢。

```
select query, substring(path,0,40) as path
from stl_unload_log
where query=2320
order by path;
```

```
query |             path
-------+---------------------------------------
  2320 | s3://my-bucket/venue0000_part_00
  2320 | s3://my-bucket/venue0001_part_00
  2320 | s3://my-bucket/venue0002_part_00
  2320 | s3://my-bucket/venue0003_part_00
(4 rows)
```

如果資料量很大，Amazon Redshift 可能會根據分割將檔案分為多個部分。例如：

```
venue_0000_part_00
venue_0000_part_01
venue_0000_part_02
venue_0001_part_00
venue_0001_part_01
venue_0001_part_02
...
```

下列 UNLOAD 命令包含在 select 陳述式中以引號括住的字串，因此引號會被逸出 (=\'OH\' ')。

```
unload ('select venuename, venuecity from venue where venuestate=\'OH\' ')
to 's3://mybucket/tickit/venue/ '
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

根據預設，UNLOAD 會失敗，而不會覆寫目的地儲存貯體中的現有檔案。為覆寫現有檔案 (包括資訊清單檔案)，指定 ALLOWOVERWRITE 選項。

```
unload ('select * from venue')
to 's3://mybucket/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
allowoverwrite;
```

卸載加密的資料檔案

UNLOAD 會自動使用 Amazon S3 伺服器端加密，搭配 AWS 受管加密金鑰 (SSE-S3) 來建立檔案。您也可以指定伺服器端加密搭配 AWS Key Management Service 金鑰 (SSE-KMS)，或用戶端加密搭配客戶受管金鑰。UNLOAD 不支援使用客戶管理金鑰的 Amazon S3 伺服器端加密。如需詳細資訊，請參閱使用伺服器端加密保護資料。

若要使用伺服器端加密搭配 AWS KMS 金鑰卸載至 Amazon S3，請使用 KMS_KEY_ID 參數提供金鑰 ID，如下範例所示。

```
unload ('select venuename, venuecity from venue')
to 's3://mybucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
KMS_KEY_ID '1234abcd-12ab-34cd-56ef-1234567890ab'
encrypted;
```

如果您想要提供自己的加密金鑰，您可以透過使用 UNLOAD 命令搭配 ENCRYPTED 選項，在 Amazon S3 中建立用戶端加密的資料檔案。UNLOAD 使用的封套加密程序與 Amazon S3 用戶端加密使用的封套加密程序相同。您可以使用 COPY 命令搭配 ENCRYPTED 選項，來載入加密的檔案。

運作程序如下：

1. 您會建立以 base64 編碼的 256 位元 AES 金鑰，您可以將此金鑰用做為私有加密金鑰或根對稱金鑰。
2. 您發出的 UNLOAD 命令包含根對稱金鑰和 ENCRYPTED 選項。
3. UNLOAD 產生一次性的對稱金鑰 (名為信封對稱金鑰) 與初始化向量 (IV)，其會使用此向量來加密資料。
4. UNLOAD 會使用根對稱金鑰來加密信封對稱金鑰。
5. UNLOAD 接著會在 Amazon S3 中存放加密資料檔案，並將每個檔案的加密信封金鑰和 IV 存放做為物件中繼資料。加密信封金鑰會存放做為物件中繼資料 `x-amz-meta-x-amz-key` 且 IV 會存放做為物件中繼資料 `x-amz-meta-x-amz-iv`。

如需封套加密程序的詳細資訊，請參閱使用適用於 Java 的 AWS 開發套件及 Amazon S3 進行用戶端資料加密文章。

若要卸載加密的資料檔案，請將根金鑰值新增至登入資料字串並包含 ENCRYPTED 選項。若使用 MANIFEST 選項，則也會加密資訊清單檔案。

```
unload ('select venuename, venuecity from venue')
to 's3://mybucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
manifest
encrypted;
```

若要卸載使用 GZIP 壓縮的加密資料檔案，請併入 GZIP 選項與根金鑰值和 ENCRYPTED 選項。

```
unload ('select venuename, venuecity from venue')
to 's3://mybucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted gzip;
```

若要載入加密的資料檔案，請使用相同的根金鑰值來新增 MASTER_SYMMETRIC_KEY 參數並包括 ENCRYPTED 選項。

```
copy venue from 's3://mybucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted;
```

以分隔或固定寬度的格式卸載資料

您可以分隔或固定寬度的格式來卸載資料。預設輸出是以縱線分隔 (使用 '|' 字元)。

以下範例將逗點指定為分隔符號：

```
unload ('select * from venue')
to 's3://mybucket/tickit/venue/comma'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter ',';
```

產生的輸出看起來像這樣：

```
20,Air Canada Centre,Toronto,ON,0
60,Rexall Place,Edmonton,AB,0
100,U.S. Cellular Field,Chicago,IL,40615
200,Al Hirschfeld Theatre,New York City,NY,0
240,San Jose Repertory Theatre,San Jose,CA,0
300,Kennedy Center Opera House,Washington,DC,0
...
```

若要卸載設為以 Tab 鍵分隔之檔案的相同結果，請發出下列命令：

```
unload ('select * from venue')
to 's3://mybucket/tickit/venue/tab'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
delimiter as '\t';
```

或者，您可以使用 FIXEDWIDTH 規格。此規格包含每個表格欄位和欄位寬度的識別符 (字元數)。由於 UNLOAD 命令會失敗，而不會截斷資料，因此請將寬度指定為至少該資料欄最長項目的長度。卸載固定寬度的資料的運作方式與卸載分隔資料的方式類似，差別在於產生的輸出未包含分隔字元。例如：

```
unload ('select * from venue')
to 's3://mybucket/tickit/venue/fw'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth '0:3,1:100,2:30,3:2,4:6';
```

固定寬度的輸出看起來如下：

```
20 Air Canada Centre         Toronto      ON0
60 Rexall Place              Edmonton     AB0
100U.S. Cellular Field       Chicago      IL40615
200Al Hirschfeld Theatre     New York CityNY0
240San Jose Repertory TheatreSan Jose     CA0
300Kennedy Center Opera HouseWashington   DC0
```

如需 FIXEDWIDTH 規格的詳細資訊，請參閱 UNLOAD 命令。

重新載入已卸載資料

若要重新載入卸載操作的結果，您可以使用 COPY 命令。

下列範例會顯示一個簡單的案例，其中會使用資訊清單檔案卸載、截斷和重新載入 VENUE 表格。

```
unload  ('select * from venue order by venueid')
to 's3://mybucket/tickit/venue/reload_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
delimiter '|';

truncate venue;

copy venue
from 's3://mybucket/tickit/venue/reload_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
```

```
delimiter '|';
```

重新載入後，VENUE 表格看起來如下：

```
select * from venue order by venueid limit 5;

 venueid |         venuename          |  venuecity  | venuestate | venueseats
---------+----------------------------+-------------+------------+-----------
       1 | Toyota Park                | Bridgeview  | IL         |          0
       2 | Columbus Crew Stadium      | Columbus    | OH         |          0
       3 | RFK Stadium                | Washington  | DC         |          0
       4 | CommunityAmerica Ballpark | Kansas City | KS         |          0
       5 | Gillette Stadium           | Foxborough  | MA         |      68756
(5 rows)
```

建立使用者定義的函數

您可以使用 SQL SELECT 子句或 Python 程式，建立自訂的純量使用者定義函數 (UDF)。新函數儲存於資料庫，可供具有足夠權限的任何使用者執行。執行自訂純量 UDF 的方式與執行現有 Amazon Redshift 函數的方式大致相同。

若為 Python UDF，除了使用標準 Python 功能外，您還可以匯入自己的自訂 Python 模組。如需詳細資訊，請參閱 UDF 的 Python 語言支援。請注意，Python 3 不適用於 Python UDF。要獲得對 Amazon Redshift UDF 的 Python 3 支持，請改用建立純量 Lambda UDF。

您也可以建立使用 Lambda 中定義的自訂函數做為 SQL 查詢一部分的 AWS Lambda UDF。Lambda UDF 可讓您撰寫複雜的 UDF，並與協力廠商元件整合。它們還可以幫助您克服目前 Python 和 SQL UDF 的一些限制。例如，它們可以協助您存取網路和儲存資源，並撰寫更完整的 SQL 陳述式。您可以使用任何由 Lambda 支援的程式設計語言 (例如 Java、圍棋、Node.js、C# PowerShell、Python 和紅寶石) 建立 Lambda UDF。或者，您也可以使用自訂執行期。

依預設，所有使用者皆可執行 UDF。如需權限的相關資訊，請參閱 UDF 安全與權限。

主題

UDF 安全與權限

若要建立 UDF，您必須具有 SQL 或 plpythonu (Python) 的語言使用權許可。根據預設，USAGE ON LANGUAGE SQL 是授予 PUBLIC，不過，您必須將 USAGE ON LANGUAGE PLPYTHONU 明確授予特定使用者或群組。

若要撤銷 SQL 的使用權，請先從 PUBLIC 撤銷使用權。然後僅將 SQL 使用權授予獲得許可建立 SQL UDF 的特定使用者或群組。下列範例撤銷從 PUBLIC 使用 SQL。然後，它會將使用權授予使用者群組 `udf_devs`。

```
revoke usage on language sql from PUBLIC;
grant usage on language sql to group udf_devs;
```

若要執行 UDF，您必須具有對每個函數執行此操作的許可。根據預設，新 UDF 的執行許可會授予 PUBLIC。若要限制使用權，請從 PUBLIC 撤銷函數的此項許可。然後將許可授予特定個人或群組。

下列範例撤銷從 PUBLIC 執行函數 `f_py_greater`。然後，它會將使用權授予使用者群組 `udf_devs`。

```
revoke execute on function f_py_greater(a float, b float) from PUBLIC;
grant execute on function f_py_greater(a float, b float) to group udf_devs;
```

根據預設，超級使用者具備所有權限。

如需詳細資訊，請參閱 GRANT 及 REVOKE。

建立純量 SQL UDF

純量 SQL UDF 包含呼叫函數時執行的 SQL SELECT 子句，並傳回單一值。CREATE FUNCTION 命令定義下列參數：

- (選用) 輸入引數。每個引數必須具有一個資料類型。
- 一個傳回資料類型。
- 一個 SQL SELECT 子句。在 SELECT 子句中，根據函數定義中的引數順序，使用 $1、$2 等來提及輸入引數。

輸入和傳回資料類型可以是任何標準 Amazon Redshift 資料類型。

不要在 SELECT 子句中包含 FROM 子句。反之，在呼叫 SQL UDF 的 SQL 陳述式中包含 FROM 子句。

SELECT 子句不可包含下列任何類型的子句：

- FROM
- INTO
- WHERE

- GROUP BY
- ORDER BY
- LIMIT

純量 SQL 函數範例

下列範例會建立一個函數，比較兩個數字並傳回較大的值。如需詳細資訊，請參閱 CREATE FUNCTION。

```
create function f_sql_greater (float, float)
  returns float
stable
as $$
  select case when $1 > $2 then $1
    else $2
  end
$$ language sql;
```

下列查詢會呼叫新的 f_sql_greater 函數來查詢 SALES 資料表，並傳回 COMMISSION 或 20% 的 PRICEPAID，以較大者為準。

```
select f_sql_greater(commission, pricepaid*0.20) from sales;
```

命名 UDF

您可以在實作之前考慮 UDF 命名慣例，來避免產生潛在的衝突和非預期的結果。因為函數名稱可以過載，所以它們可能與現有和未來的 Amazon Redshift 函數名稱發生衝突。本主題討論過載，並說明避免衝突的策略。

多載函數名稱

函數是以其名稱和簽章來識別，而簽章是輸入引數的數目和引數的資料類型。如果相同結構描述中的兩個函數具有不同的簽章，則它們可以具有相同的名稱。換言之，函數名稱可以過載。

當您執行查詢時，查詢引擎會根據您提供的引數數目和引數的資料類型來決定要呼叫哪個函數。您可以使用過載來模擬具有可變引數數目的函數，而此數目最多可為 CREATE FUNCTION 命令允許的限制。

防止 UDF 命名衝突

我們建議您在所有 UDF 名稱前加上 f_。Amazon Redshift 建議您使用字首 f_ 來命名所有 UDF，藉由以 f_ 做為 UDF 名稱的字首，您可以確保 UDF 名稱不會與任何現有或未來的 Amazon Redshift 內建 SQL 函數名稱發生衝突。例如，藉由將新的 UDF 命名為 f_sum，您可以避免與 Amazon Redshift SUM 函數發生衝突。同樣地，如果您將新函數命名為 f_fibonacci，則可在 Amazon Redshift 於未來版本中新增名為 FIBONACCI 的函數時避免發生衝突。

如果 UDF 和內建函數存在於不同的結構描述中，則您可以使用與現有 Amazon Redshift 內建 SQL 函數相同的名稱和簽章來建立 UDF，而函數名稱不會過載。因為內建函數存在於系統目錄結構描述 (pg_catalog) 中，所以您可以在另一個結構描述 (例如公有或使用者定義的結構描述) 中建立名稱相同的 UDF。在某些情況下，您可能會呼叫未使用結構描述名稱明確限定的函數。如果是這樣，Amazon Redshift 預設會先搜尋 pg_catalog 結構描述。因此，內建函數會在具有相同名稱的新 UDF 之前執行。

您可以透過將搜尋路徑設定為將 pg_catalog 放在末尾來變更此行為。如果您這樣做，您的 UDF 會優先於內建函數，但這種做法可能會導致非預期的結果。採用唯一命名策略 (例如使用保留字首 f_) 是更為可靠的實務。如需詳細資訊，請參閱 SET 及 search_path。

建立純量 Python UDF

純量 Python UDF 包含呼叫函數時執行的 Python 程式，並傳回單一值。CREATE FUNCTION 命令定義下列參數：

- (選用) 輸入引數。每個引數必須具有一個名稱和一個資料類型。
- 一個傳回資料類型。
- 一個可執行的 Python 程式。

這時的輸入及傳回的資料類型可為 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE PRECISION、BOOLEAN、CHAR、VARCHAR、DATE 或 TIMESTAMP。此外，Python UDF 還可以使用資料類型 ANYELEMENT，而 Amazon Redshift 會根據執行期提供的引數，自動將其轉換為標準資料類型。如需更多資訊，請參閱 ANYELEMENT 資料類型

當 Amazon Redshift 查詢呼叫純量 UDF 時，下列步驟會在執行期發生。

1. 函數會將輸入引數轉換為 Python 資料類型。

   如需將 Amazon Redshift 資料類型映射至 Python 資料類型，請參閱 Python UDF 資料類型。
2. 函數會執行 Python 程式，剖析已轉換的輸入引數。

3. Python 程式碼會傳回單一值。傳回值的資料類型必須對應至函數定義所指定的 RETURNS 資料類型。
4. 函數會將 Python 傳回值轉換為指定的 Amazon Redshift 資料類型，然後將該值傳回至佇列。

> **Note**
>
> Python 3 不可用於 Python UDF。要獲得對 Amazon Redshift UDF 的 Python 3 支持，請改用建立純量 Lambda UDF。

純量 Python UDF 範例

下列範例會建立一個函數，比較兩個數字並傳回較大的值。請注意，雙貨幣符號 ($$) 之間程式碼的縮排是 Python 需求。如需詳細資訊，請參閱 CREATE FUNCTION。

```
create function f_py_greater (a float, b float)
  returns float
stable
as $$
  if a > b:
    return a
  return b
$$ language plpythonu;
```

下列查詢會呼叫新的 f_greater 函數來查詢 SALES 資料表，並傳回 COMMISSION 或 20% 的 PRICEPAID，以較大者為準。

```
select f_py_greater (commission, pricepaid*0.20) from sales;
```

Python UDF 資料類型

Python UDF 可對輸入引數和函數的傳回值使用任何標準 Amazon Redshift 資料類型。除了標準資料類型外，UDF 還支援資料類型 ANYELEMENT，而 Amazon Redshift 會根據執行期提供的引數自動將其轉換為標準資料類型。純量 UDF 可以傳回資料類型 ANYELEMENT。如需詳細資訊，請參閱 ANYELEMENT 資料類型。

在執行期間，Amazon Redshift 會將引數從 Amazon Redshift 資料類型轉換為 Python 資料類型以進行處理。然後，它將傳回值從 Python 資料類型轉換為對應的 Amazon Redshift 資料類型。如需 Amazon Redshift 資料類型的相關資訊，請參閱資料類型。

下表將 Amazon Redshift 資料類型映射至 Python 資料類型。

Amazon Redshift 資料類型	Python 資料類型
smallint integer bigint short 長整數	int
decimal 或 numeric	decimal
double real	float
布林值	bool
char varchar	string
timestamp	datetime

ANYELEMENT 資料類型

ANYELEMENT 是多型資料類型。這表示，若對引數的資料類型使用 ANYELEMENT 來宣告函數，則函數可以接受任何標準 Amazon Redshift 資料類型，做為呼叫函數時該引數的輸入。ANYELEMENT 引數會設為呼叫函數時實際傳遞至其中的資料類型。

如果函數使用多個 ANYELEMENT 資料類型，則在呼叫函數時，它們必須全都解析為相同的實際資料類型。所有 ANYELEMENT 引數資料類型都會設為傳遞至 ANYELEMENT 之第一個引數的實際資料類型。例如，宣告為 `f_equal(anyelement, anyelement)` 的函數將採用任何兩個輸入值，只要它們是相同的資料類型。

如果函數的輸入值宣告為 ANYELEMENT，則至少一個輸入引數必須是 ANYELEMENT。傳回值的實際資料類型將與提供給 ANYELEMENT 輸入引數的實際資料類型相同。

UDF 的 Python 語言支援

您可以根據 Python 程式設計語言建立自訂 UDF。Python 2.7 standard library 可供 UDF 使用，但下列模組除外：

- ScrolledText
- Tix
- Tkinter
- tk
- turtle
- smtpd

除了 Python 標準程式庫外，下列模組是 Amazon Redshift 實作的一部分：

- numpy 1.8.2
- pandas 0.14.1
- python-dateutil 2.2
- pytz 2014.7
- scipy 0.12.1
- six 1.3.0
- wsgiref 0.1.2

您也可以匯入自己的自訂 Python 模組，並執行 CREATE LIBRARY 命令，使它們可供 UDF 使用。如需詳細資訊，請參閱 匯入自訂 Python 程式庫模組。

Important

Amazon Redshift 會透過 UDF 全面封鎖檔案系統的網路存取和寫入存取。

Note

Python 3 不可用於 Python UDF。要獲得對 Amazon Redshift UDF 的 Python 3 支持，請改用建立純量 Lambda UDF。

匯入自訂 Python 程式庫模組

您可以使用 Python 語言語法來定義純量函數。您可以使用 Python 標準程式庫模組和 Amazon Redshift 預先安裝模組。您也可以建立自己的自訂 Python 程式庫模組，並將程式庫匯入叢集，或使用 Python 或第三方的現有程式庫。

您建立的程式庫不可包含與 Python 標準程式庫模組或 Amazon Redshift 預先安裝 Python 模組同名的模組。如果現有使用者安裝的程式庫使用相同的 Python 套件做為您建立的程式庫，則您必須捨棄現有的程式庫，然後才能安裝新的程式庫。

您必須是超級使用者或具有 USAGE ON LANGUAGE plpythonu 權限，才能安裝自訂程式庫；不過，任何具有足夠權限來建立函數的使用者都可以使用已安裝的程式庫。您可以查詢 PG_LIBRARY 系統目錄，來檢視叢集上已安裝之程式庫的相關資訊。

將自訂 Python 模組匯入您的叢集

本節提供將自訂 Python 模組匯入至您的叢集的範例。若要執行本節中的步驟，您必須具有 Amazon S3 儲存貯體，而您會在其中上傳程式庫套件。然後，在您的叢集中安裝套件。如需建立儲存貯體的相關資訊，請前往《Amazon Simple Storage Service 使用者指南》中的建立儲存貯體。

在此範例中，讓我們假設您建立 UDF，來使用資料中的位置和距離。從 SQL 用戶端工具連接至您的 Amazon Redshift 叢集，並執行下列命令來建立函數。

```
CREATE FUNCTION f_distance (x1 float, y1 float, x2 float, y2 float) RETURNS float
 IMMUTABLE as $$
    def distance(x1, y1, x2, y2):
        import math
        return math.sqrt((y2 - y1) ** 2 + (x2 - x1) ** 2)

    return distance(x1, y1, x2, y2)
$$ LANGUAGE plpythonu;

CREATE FUNCTION f_within_range (x1 float, y1 float, x2 float, y2 float) RETURNS bool
 IMMUTABLE as $$
    def distance(x1, y1, x2, y2):
        import math
        return math.sqrt((y2 - y1) ** 2 + (x2 - x1) ** 2)

    return distance(x1, y1, x2, y2) < 20
$$ LANGUAGE plpythonu;
```

請注意，會複製先前函數中的幾行程式碼。需要進行此複製的原因是 UDF 無法參考另一個 UDF 的內容，而且這兩個函數都需要相同功能。然而，不是複製多個函數中的程式碼，而是您可以建立自訂程式庫，並設定您的函數來使用它。

若要這樣做，首先遵循下列步驟來建立程式庫套件：

1. 建立名為 geometry 的資料夾。此資料夾是程式庫的最上層套件。
2. 在 geometry 資料夾中，建立名為 `__init__.py` 的檔案。請注意，檔案名稱包含兩個雙底線字元。此檔案向 Python 表示可以初始化套件。
3. 另外，在 geometry 資料夾中，建立名為 trig 的資料夾。此資料夾是程式庫的子套件。
4. 在 trig 資料夾中，建立另一個名為 `__init__.py` 的檔案，以及一個名為 `line.py` 的檔案。在此資料夾中，`__init__.py` 向 Python 表示可以初始化子套件，而且 `line.py` 是包含程式庫程式碼的檔案。

   您的資料夾和檔案結構應該與下列資料夾和檔案結構相同：

   ```
   geometry/
      __init__.py
      trig/
         __init__.py
         line.py
   ```

   如需套件結構的相關資訊，請前往 Python 網站上 Python 教學課程中的模組。
5. 下列程式碼包含程式庫的類別和成員函數。複製它並貼至 `line.py`。

   ```
   class LineSegment:
     def __init__(self, x1, y1, x2, y2):
       self.x1 = x1
       self.y1 = y1
       self.x2 = x2
       self.y2 = y2
     def angle(self):
       import math
       return math.atan2(self.y2 - self.y1, self.x2 - self.x1)
     def distance(self):
       import math
       return math.sqrt((self.y2 - self.y1) ** 2 + (self.x2 - self.x1) ** 2)
   ```

在您建立了套件之後，請執行下列動作，來準備套件並將其上傳至 Amazon S3。

1. 將 geometry 資料夾的內容壓縮成名為 geometry.zip 的 .zip 檔案。不要包括 geometry 資料夾本身；只包括資料夾的內容，如下所示：

```
geometry.zip
   __init__.py
   trig/
      __init__.py
      line.py
```

2. 將 geometry.zip 上傳至您的 Amazon S3 儲存貯體。

> **Important**
>
> 如果 Amazon S3 儲存貯體不是位於與 Amazon Redshift 叢集所在的同一區域，您必須使用 REGION 選項來指定資料所在的區域。如需詳細資訊，請參閱 CREATE LIBRARY。

3. 從您的 SQL 用戶端工具中，執行下列命令來安裝程式庫。<bucket_name><access key id><secret key>取代為值區的名稱，並取代為 AWS Identity and Access Management (IAM) 使用者登入資料中的存取金鑰和秘密存取金鑰。

```
CREATE LIBRARY geometry LANGUAGE plpythonu FROM 's3://<bucket_name>/geometry.zip'
 CREDENTIALS 'aws_access_key_id=<access key id>;aws_secret_access_key=<secret key>';
```

在您的叢集中安裝程式庫之後，您需要設定函數來使用程式庫。若要這樣做，請執行下列命令。

```
CREATE OR REPLACE FUNCTION f_distance (x1 float, y1 float, x2 float, y2 float) RETURNS
 float IMMUTABLE as $$
    from trig.line import LineSegment

    return LineSegment(x1, y1, x2, y2).distance()
$$ LANGUAGE plpythonu;

CREATE OR REPLACE FUNCTION f_within_range (x1 float, y1 float, x2 float, y2 float)
 RETURNS bool IMMUTABLE as $$
    from trig.line import LineSegment

    return LineSegment(x1, y1, x2, y2).distance() < 20
$$ LANGUAGE plpythonu;
```

在上述命令中，`import trig/line` 會從本節中的原始函數中刪除重複的程式碼。您可以在多個 UDF 中使用此程式庫所提供的功能。請注意，若要匯入模組，您只需要指定子套件和模組名稱的路徑 (`trig/line`)。

UDF 限制條件

在本主題中列出的限制內，只要可以使用 Amazon Redshift 內建純量函數之處，就可以使用 UDF。如需詳細資訊，請參閱 SQL 函數參考。

Amazon Redshift Python UDF 有下列限制：

- Python UDF 無法存取網路，也無法讀取或寫入至檔案系統。
- 使用者安裝的 Python 程式庫的總和大小不得超過 100 MB。
- 每個叢集可以同時執行的 Python UDF 數目限制為叢集的總計並行層級的四分之一。例如，如果設定並行數為 15 的叢集，則最多可有三個 UDF 同時執行。在達到限制之後，UDF 會排入工作負載管理佇列內等待執行。SQL UDF 沒有並行限制。如需詳細資訊，請參閱 實作工作負載管理。
- 使用 Python UDF 時，Amazon Redshift 不支援 SUPER 和 HLLSKETCH 資料類型。

記錄 UDF 中的錯誤和警告

您可以使用 Python 記錄模組，在 UDF 中建立使用者定義的錯誤和警告訊息。您可以在查詢執行之後，查詢 SVL_UDF_LOG 系統檢視來擷取日誌記錄的訊息。

> **Note**
>
> UDF 記錄會使用叢集資源，因此可能會影響系統效能。我們建議只針對開發和故障診斷實作記錄。

在查詢執行期間，日誌處理常式會將訊息連同對應的函數名稱、節點和配量一起寫入至 SVL_UDF_LOG 系統檢視。日誌處理常式會在每個配量每則訊息寫入一個資料列至 SVL_UDF_LOG。訊息會截斷至 4096 個位元組。UDF 日誌限制為每個配量 500 個資料列。當日誌滿時，日誌處理常式會捨棄較舊訊息，並將警告訊息新增至 SVL_UDF_LOG。

> **Note**
>
> Amazon Redshift UDF 日誌處理常式會將換行 (\n)、縱線 (|) 字元和反斜線 (\) 字元加上反斜線 (\)，使它們逸出。

依預設，UDF 日誌層級會設為 WARNING。日誌層級為 WARNING、ERROR 和 CRITICAL 的訊息都會加以記錄。具有較低嚴重性 INFO、DEBUG 和 NOTSET 的訊息則會略過。若要測試 UDF 日誌層級，請使用 Python 記錄器方法。例如，下列程式碼會將日誌層級設為 INFO。

```
logger.setLevel(logging.INFO)
```

如需使用 Python 記錄模組的相關資訊，請參閱 Python 文件中的 Python 的記錄機能。

下列範例會建立名為 f_pyerror 的函數，用來匯入 Python 記錄模組、將記錄器執行個體化，以及記錄錯誤。

```
CREATE OR REPLACE FUNCTION f_pyerror()
RETURNS INTEGER
VOLATILE AS
$$
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
logger.info('Your info message here')
return 0
$$ language plpythonu;
```

下列範例會查詢 SVL_UDF_LOG，來檢視前一個範例中記錄的訊息。

```
select funcname, node, slice, trim(message) as message
from svl_udf_log;

  funcname  | query | node | slice |   message
------------+-------+------+-------+-----------------
  f_pyerror | 12345 |     1|     1 | Your info message here
```

建立純量 Lambda UDF

Amazon Redshift 可以使用中定義的自訂函數 AWS Lambda 做為 SQL 查詢的一部分。
您可以使用 Lambda 支援的任何程式設計語言撰寫純量 Lambda UDF，例如 Java、圍棋 PowerShell、Node.js、C#、Python 和紅寶石。或者，您也可以使用自訂執行期。

Lambda UDF 是在 Lambda 中定義和管理的，您可以控制存取權限，以便在 Amazon Redshift 中調用這些 UDF。您可以在同一個查詢中調用多個 Lambda 函數，或多次調用相同的函數。

在支援純量函數的 SQL 陳述式的任何子句中使用 Lambda UDF。您也可以在任何 SQL 陳述式 (例如 SELECT、UPDATE、INSERT 或 DELETE) 中使用 Lambda UDF。

> **Note**
>
> 使用 Lambda UDF 可能會產生來自 Lambda 服務的額外費用。是否如此取決於 Lambda 請求 (UDF 調用) 的數量和 Lambda 程式執行的總持續時間等因素。但是，在 Amazon Redshift 中使用 Lambda UDF 無需額外收費。如需 AWS Lambda 定價的相關資訊，請參閱AWS Lambda 定價。
>
> Lambda 請求的數目會根據使用 Lambda UDF 的特定 SQL 陳述式子句而有所不同。例如，假設函數用於 WHERE 子句中，如下所示。
>
> ```
> SELECT a, b FROM t1 WHERE lambda_multiply(a, b) = 64; SELECT a, b
> FROM t1 WHERE a*b = lambda_multiply(2, 32)
> ```
>
> 在此情況下，Amazon Redshift 會針對每個陳述式呼叫第一個 SELECT 陳述式，並只呼叫第二個 SELECT 陳述式一次。
>
> 不過，在查詢的投影部分使用 UDF 可能只會針對結果集中的每個限定或彙總資料列調用 Lambda 函數一次。

註冊 Lambda UDF

CREATE EXTERNAL FUNCTION 命令會建立下列參數：

- (選用) 具有資料類型的引數清單。
- 一個傳回資料類型。
- 由 Amazon Redshift 呼叫之外部函數的一個函數名稱。
- Amazon Redshift 叢集有權擔任和呼叫 Lambda 的一個 IAM 角色。
- Lambda UDF 調用的一個 Lambda 函數名稱。

如需 CREATE EXTERNAL FUNCTION 的相關資訊，請參閱 CREATE EXTERNAL FUNCTION。

此函數的輸入和傳回資料類型可以是任何標準 Amazon Redshift 資料類型。

Amazon Redshift 可確保外部函數可以傳送和接收批次引數和結果。

管理 Lambda UDF 安全和權限

若要建立 Lambda UDF，請確定您擁有使用 LANGUAGE EXFUNC 的許可。您必須對特定使用者、群組或公眾明確授予 USAGE ON LANGUAGE EXFUNC 或撤銷 USAGE ON LANGUAGE EXFUNC 的使用。

下列範例將 EXFUNC 的使用權授予 PUBLIC。

```
grant usage on language exfunc to PUBLIC;
```

下列範例會從 PUBLIC 撤銷 exfunc 的使用權，然後將使用權授予使用者群組 lambda_udf_devs。

```
revoke usage on language exfunc from PUBLIC;
grant usage on language exfunc to group lambda_udf_devs;
```

若要執行 Lambda UDF，請確認您具有每個所呼叫函數的許可。根據預設，新 Lambda UDF 的執行許可會授予 PUBLIC。若要限制使用權，請從 PUBLIC 撤銷函數的此項許可。然後將權限授予特定使用者或群組。

下列範例撤銷從 PUBLIC 執行函數 exfunc_sum。然後，它會將使用權授予使用者群組 lambda_udf_devs。

```
revoke execute on function exfunc_sum(int, int) from PUBLIC;
grant execute on function exfunc_sum(int, int) to group lambda_udf_devs;
```

根據預設，超級使用者具備所有權限。

如需授予和撤銷權限的相關資訊，請參閱 GRANT 和 REVOKE。

設定 Lambda UDF 的授權參數

CREATE EXTERNAL FUNCTION 命令需要授權才能在 AWS Lambda 中調用 Lambda 函數。若要啟動授權，請在執行建立外部函數命令時指定 AWS Identity and Access Management (IAM) 角色。如需 IAM 角色的相關資訊，請參閱《IAM 使用者指南》中的 IAM 角色。

如果現有 IAM 角色有權調用連接到叢集的 Lambda 函數，則您可以將命令的 IAM_ROLE 參數中的角色 Amazon Resource Name (ARN) 替換為您的角色。以下各節說明在 CREATE EXTERNAL FUNCTION 命令中使用 IAM 角色的步驟。

建立適用於 Lambda 的 IAM 角色

IAM 角色需要許可才能調用 Lambda 函數。建立 IAM 角色時，請使用下列其中一種方式提供許可：

- 建立 IAM 角色時，請在連接許可政策頁面上連接 AWSLambdaRole 政策。AWSLambdaRole 政策授予調用 Lambda 函數的許可，這是最低要求。如需詳細資訊和其他政策，請參閱《AWS Lambda 開發人員指南》中的 AWS Lambda 的身分型 IAM 政策。
- 建立您自己的自訂政策，並使用具有該函數 ARN 之所有資源或特定 Lambda 函數的 lambda:InvokeFunction 許可，連接到您的 IAM 角色。如需如何建立政策的相關資訊，請參閱《IAM 使用者指南》中的建立 IAM 政策。

  下列範例政策可讓您在特定 Lambda 函數上調用 Lambda。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Invoke",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-west-2:123456789012:function:my-function"
        }
    ]
}
```

  如需 Lambda 函數資源的相關資訊，請參閱《IAM API 參考》中的 Lambda 動作的資源和條件。

  使用所需許可建立自訂政策後，您可以在建立 IAM 角色時，在連接許可政策頁面上將政策連接到 IAM 角色。

如需建立 IAM 角色的步驟，請參閱 Amazon Redshift 管理指南中的授權 Amazon Redshift 代表您存取其他 AWS 服務。

如果您不要建立新的 IAM 角色，可以將先前提到的許可新增至現有的 IAM 角色。

將 IAM 角色與叢集建立關聯

將 IAM 角色連接至您的叢集。您可以使用 Amazon Redshift 管理主控台、CLI 或 API 將角色新增至叢集，或檢視與叢集相關聯的角色。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的將 IAM 角色與叢集建立關聯。

在命令中包含 IAM 角色

在 CREATE EXTERNAL FUNCTION 命令中包含 IAM 角色 ARN。建立 IAM 角色時，IAM 會傳回角色的 Amazon Resource Name (ARN)。若要指定 IAM 角色，請使用 `IAM_ROLE` 參數來提供角色 ARN。以下顯示 `IAM_ROLE` 參數的語法。

```
IAM_ROLE 'arn:aws:iam::aws-account-id:role/role-name'
```

若要調用位於相同區域內其他帳戶的 Lambda 函數，請參閱在 Amazon Redshift 中鏈結 IAM 角色。

在 Amazon Redshift 和 AWS Lambda 之間使用 JSON 介面

Amazon Redshift 對 Amazon Redshift 進行通訊的所有 Lambda 函數使用通用介面。

下表顯示您可預期用於 JSON 承載之指定 Lambda 函數的輸入欄位清單。

欄位名稱	描述	值範圍
request_id	全域唯一識別碼 (UUID)，可唯一識別每個調用請求。	有效的 UUID。
叢集	叢集的完整 Amazon Resource Name (ARN)。	有效的叢集 ARN。
使用者	進行呼叫的使用者名稱。	有效的使用者名稱。
database	執行查詢之資料庫的名稱。	有效的資料庫名稱。
external_function	進行呼叫之外部函數的完整名稱。	有效的完整函數名稱。

欄位名稱	描述	值範圍
query_id	進行呼叫之查詢的查詢 ID。	有效的查詢 ID。
num_records	承載中的引數數目。	值為 1 - 2^64。
引數	指定格式的資料承載。	陣列格式的資料必須是 JSON 陣列。如果引數數目大於 1，則每個元素都是一個記錄，該記錄是一個陣列。透過使用陣列，Amazon Redshift 會保留承載中記錄的順序。

JSON 陣列的順序決定批次處理的順序。Lambda 函數必須反覆處理引數，並產生確切的記錄數。以下是承載的範例。

```
{
  "request_id" : "23FF1F97-F28A-44AA-AB67-266ED976BF40",
  "cluster" : "arn:aws:redshift:xxxx",
  "user" : "adminuser",
  "database" : "db1",
  "external_function": "public.foo",
  "query_id" : 5678234,
  "num_records" : 4,
  "arguments" : [
     [ 1, 2 ],
     [ 3, null],
     null,
     [ 4, 6]
   ]
 }
```

Lambda 函數的傳回輸出包含下列欄位。

欄位名稱	描述	值範圍
success	函數成功或失敗的指示。	值為 "true" 或 "false"。

欄位名稱	描述	值範圍
error_msg	如果成功值為 "false" (如果函數失敗)，則會顯示錯誤訊息；否則，會忽略此欄位。	有效的訊息。
num_records	承載中的記錄數。	值為 1 - 2^64。
results	以指定格式呼叫的結果。	N/A

以下是 Lambda 函數輸出的範例。

```
{
  "success": true,   // true indicates the call succeeded
  "error_msg" : "my function isn't working",  // shall only exist when success != true
  "num_records": 4,      // number of records in this payload
  "results" : [
     1,
     4,
     null,
     7
   ]
}
```

當您從 SQL 查詢呼叫 Lambda 函數時，Amazon Redshift 可確保連線的安全性，並考量下列事項：

- GRANT 和 REVOKE 許可。如需 UDF 安全和權限的相關資訊，請參閱UDF 安全與權限。
- Amazon Redshift 只會向指定的 Lambda 函數提交最小資料集。
- Amazon Redshift 只會使用指定的 IAM 角色呼叫指定的 Lambda 函數。

使用者定義函數 (UDF) 的使用範例

您可以將 Amazon Redshift 與其他元件整合，以使用使用者定義的函數來解決業務問題。以下是其他人如何將 UDF 用於其使用案例的一些範例：

- 使用 Amazon Redshift Lambda UDF 存取外部元件 - 說明 Amazon Redshift Lambda UDF 如何運作以及逐步建立 Lambda UDF 的過程。
- 透過 Amazon Redshift、Amazon Translate 和 Amazon Comprehend 使用 SQL 函數翻譯和分析文字 - 提供預先建置的 Amazon Redshift Lambda UDF，您只需點擊幾下即可安裝這些 UDF，以翻譯、編輯和分析文字欄位。
- 從 Amazon Redshift 存取 Amazon Location Service - 說明如何使用 Amazon Redshift Lambda UDF 與 Amazon Location Service 整合。
- 使用 Amazon Redshift 和 Protegrity 進行資料記號化 - 說明如何將 Amazon Redshift Lambda UDF 與 Protegrity Serverless 產品整合。
- Amazon Redshift UDF - Amazon Redshift SQL、Lambda 和 Python UDF 的集合。

在 Amazon Redshift 中建立預存程序

您可以使用 PostgreSQL 程序性語言 PL/pgSQL 來定義 Amazon Redshift 預存程序，以執行一組 SQL 查詢和邏輯操作。程序存放於資料庫，可供具有足夠資料庫權限的任何使用者執行。

與使用者定義的函數 (UDF) 不同，除了 SELECT 查詢，預存程序還可以結合資料定義語言 (DDL) 和資料操作語言 (DML)。預存程序不需要傳回任何值。您可以使用程序性語言 (包括迴圈和條件式表達式) 來控制邏輯流量。

如需用於建立和管理預存程序的 SQL 命令的詳細資訊，請參閱下列命令主題：

- CREATE PROCEDURE
- ALTER PROCEDURE
- DROP PROCEDURE
- SHOW PROCEDURE
- CALL
- GRANT
- REVOKE
- ALTER DEFAULT PRIVILEGES

主題

- Amazon Redshift 中的預存程序概觀
- PL/pgSQL 語言參考

Amazon Redshift 中的預存程序概觀

預存程序通常用於封裝資料轉換、資料驗證的邏輯和特定商業邏輯。將多個 SQL 步驟結合在一個預存程序中，可以減少應用程式和資料庫之間的往返次數。

如需精細的存取控制，您可以建立預存程序來執行功能，而不需要讓使用者存取基礎資料表。例如，只有擁有者或超級使用者才能截斷資料表，使用者需要寫入權限，才能將資料插入資料表。您可以建立預存程序來執行任務，而不是授予使用者對基礎資料表的權限。然後，您會給予使用者權限來執行預存程序。

具有 DEFINER 安全屬性的預存程序會以預存程序擁有者的權限執行。根據預設，預存程序具有 INVOKER 安全性，這表示程序會使用程序呼叫使用者的權限。

若要建立預存程序，請使用 CREATE PROCEDURE 命令。若要執行程序，請使用 CALL 命令。本節稍後提供範例。

> **Note**
>
> 在建立 Amazon Redshift 預存程序時，有些用戶端可能會顯示下列錯誤。
>
> ```
> ERROR: 42601: [Amazon](500310) unterminated dollar-quoted string at or near "$$
> ```
>
> 導致此錯誤的原因是用戶端無法正確剖析含有分號分隔陳述式和貨幣符號 ($) 引用的 CREATE PROCEDURE 陳述式。這會導致只有一部分陳述式傳送至 Amazon Redshift 伺服器。您通常可以透過用戶端的 `Run as batch` 或 `Execute selected` 選項來解決此錯誤。
> 舉例而言，使用 Aginity 用戶端時，您可以利用 `Run entire script as batch` 選項。建議您使用 SQL Workbench/J 版本 124。使用 SQL Workbench/J 版本 125 時，請考慮指定替代分隔符號來解決這個問題。
> CREATE PROCEDURE 包含以分號 (;) 分隔的 SQL 陳述式。定義替代分隔符號 (例如斜線 (/))，並將其放在 CREATE PROCEDURE 陳述式的結尾，會將陳述式傳送至 Amazon Redshift 伺服器進行處理。以下是範例。
>
> ```
> CREATE OR REPLACE PROCEDURE test()
> AS $$
> BEGIN
>   SELECT 1 a;
> END;
> $$
> LANGUAGE plpgsql
> ;
> /
> ```
>
> 如需詳細資訊，請參閱 SQL Workbench/J 文件中的 Alternate delimiter (替代分隔符號)。或者使用具有更好支援剖析 CREATE 程序陳述式的用戶端，例如 Amazon Redshift 主控台中的查詢編輯器或 TablePlus.

主題

- 命名預存程序

- 預存程序的安全和權限
- 傳回結果集
- 管理交易
- 捕捉錯誤
- 記錄預存程序日誌
- 預存程序支援的考量

下列範例顯示不具有輸出引數的程序。根據預設，引數是輸入 (IN) 引數。

```
CREATE OR REPLACE PROCEDURE test_sp1(f1 int, f2 varchar)
AS $$
BEGIN
  RAISE INFO 'f1 = %, f2 = %', f1, f2;
END;
$$ LANGUAGE plpgsql;

call test_sp1(5, 'abc');
INFO: f1 = 5, f2 = abc
CALL
```

> **Note**
>
> 當您撰寫預存程序時，我們建議您採用保護敏感值的最佳作法：
> 不要在預存程序邏輯中對任何敏感資訊進行硬式編碼。例如，請勿在預存程序主體的 CREATE USER 陳述式中指派使用者密碼。這會造成安全性風險，因為硬式編碼值可以記錄為目錄資料表中的結構描述資料。應改為透過參數將敏感值 (例如密碼) 當做引數傳遞給預存程序。
> 如需預存程序的詳細資訊，請參閱 CREATE PROCEDURE 和在 Amazon Redshift 中建立預存程序。如需有關目錄資料表的詳細資訊，請參閱 系統目錄資料表。

下列範例顯示具有輸出引數的程序。引數為輸入 (IN)、輸入和輸出 (INOUT) 及輸出 (OUT)。

```
CREATE OR REPLACE PROCEDURE test_sp2(f1 IN int, f2 INOUT varchar(256), out_var OUT
 varchar(256))
AS $$
DECLARE
  loop_var int;
BEGIN
```

```
  IF f1 is null OR f2 is null THEN
    RAISE EXCEPTION 'input cannot be null';
  END IF;
  DROP TABLE if exists my_etl;
  CREATE TEMP TABLE my_etl(a int, b varchar);
    FOR loop_var IN 1..f1 LOOP
        insert into my_etl values (loop_var, f2);
        f2 := f2 || '+' || f2;
    END LOOP;
  SELECT INTO out_var count(*) from my_etl;
END;
$$ LANGUAGE plpgsql;


call test_sp2(2,'2019');

         f2           | column2
----------------------+---------
 2019+2019+2019+2019 | 2
(1 row)
```

命名預存程序

如果您定義的程序有相同名稱和不同的引數資料類型 (或簽章)，則會建立新程序。因此，程序名稱會多載。如需詳細資訊，請參閱多載程序名稱。Amazon Redshift 不會啟用根據輸出引數的程序多載。您不能有兩個程序是具有相同的名稱和輸入引數資料類型，但卻有不同的輸出引數類型。

擁有者或超級使用者可以將預存程序的主體換成具有相同簽章的預存程序。若要變更預存程序的簽章或傳回類型，請捨棄預存程序再重新建立。如需詳細資訊，請參閱 DROP PROCEDURE 及 CREATE PROCEDURE。

在實作預存程序之前，您可以考量命名慣例，以避免潛在的衝突和非預期的結果。因為您可以多載程序名稱，它們可能會與現有和未來的 Amazon Redshift 程序名稱相衝突。

多載程序名稱

程序是以其名稱和簽章來識別，而簽章是輸入引數的數目和引數的資料類型。如果相同結構描述中的兩個程序具有不同的簽章，則它們可以具有相同的名稱。換言之，您可以過載程序名稱。

當您執行程序時，查詢引擎會根據您提供的引數數目和引數的資料類型，以決定要呼叫哪個程序。您可以使用過載來模擬具有可變引數數目的程序，而此數目最多可為 CREATE PROCEDURE 命令允許的限制。如需詳細資訊，請參閱CREATE PROCEDURE。

防止命名衝突

我們建議您在所有程序名稱前加上 sp_。Amazon Redshift 專門為預存程序保留 sp_ 字首。您可以在程序名稱前方加上 sp_，以確保程序名稱不會與任何現有或未來的 Amazon Redshift 程序名稱相衝突。

預存程序的安全和權限

根據預設，所有使用者具有建立程序的權限。若要建立程序，您必須有 PL/pgSQL 語言的 USAGE 權限，此權限預設會授予 PUBLIC。根據預設，只有超級使用者和擁有者才有權限呼叫程序。如果超級使用者想要防止使用者建立預存程序，他們可以在 PL/pgSQL 上對使用者執行 REVOKE USAGE。

若要呼叫程序，您必須獲授予程序的 EXECUTE 權限。根據預設，新程序的 EXECUTE 權限會授予程序擁有者和超級使用者。如需詳細資訊，請參閱GRANT。

根據預設，建立程序的使用者就是擁有者。根據預設，擁有者具有程序的 CREATE、DROP 和 EXECUTE 權限。超級使用者具備所有權限。

SECURITY 屬性控制程序對資料物件的存取權限。當您建立預存程序時，您可以將 SECURITY 屬性設為 DEFINER 或 INVOKER。如果您指定 SECURITY INVOKER，程序會使用程序叫用使用者的權限。如果您指定 SECURITY DEFINER，程序會使用程序擁有者的權限。INVOKER 為預設值。

因為 SECURITY DEFINER 程序是以擁有使用者的權限執行，請務必確保不能誤用程序。若要確保不能誤用 SECURITY DEFINER 程序，請執行下列動作：

- 將 SECURITY DEFINER 程序的 EXECUTE 授予特定使用者，而非 PUBLIC。
- 以結構描述名稱來限定程序需要存取的所有資料庫物件。例如，使用 myschema.mytable，而不只是 mytable。
- 如果您無法以結構描述來限定物件名稱，請在建立程序時使用 SET 選項來設定 search_path。設定 search_path 來排除不受信任使用者所能撰寫的任何結構描述。此方法防止此程序的任何發起人建立物件 (例如，資料表或檢視)，而遮罩應該由程序使用的物件。如需 SET 選項的詳細資訊，請參閱 CREATE PROCEDURE。

下列範例將 search_path 設為 admin，以確保從 admin 結構描述存取 user_creds 資料表，而不是從 public 或發起人的 search_path 中任何其他結構描述來存取。

```
CREATE OR REPLACE PROCEDURE sp_get_credentials(userid int, o_creds OUT varchar)
AS $$
BEGIN
  SELECT creds INTO o_creds
```

```
  FROM user_creds
  WHERE user_id = $1;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER
-- Set a secure search_path
SET search_path = admin;
```

傳回結果集

您可以使用游標或暫存資料表來傳回結果集。

傳回游標

若要傳回游標，請使用以 refcursor 資料類型定義的 INOUT 引數建立程序。當您呼叫程序時，請為游標命名。然後，您可以依照名稱從游標中獲取結果。

下列範例建立名為 get_result_set 的程序，具有名為 rs_out 且使用 refcursor 資料類型的 INOUT 引數。此程序使用 SELECT 陳述式開啟游標。

```
CREATE OR REPLACE PROCEDURE get_result_set (param IN integer, rs_out INOUT refcursor)
AS $$
BEGIN
  OPEN rs_out FOR SELECT * FROM fact_tbl where id >= param;
END;
$$ LANGUAGE plpgsql;
```

下列 CALL 命令開啟名為 mycursor 的游標。請只在交易內使用游標。

```
BEGIN;
CALL get_result_set(1, 'mycursor');
```

開啟游標後，您就可以從游標擷取，如下列範例所示。

```
FETCH ALL FROM mycursor;

    id | secondary_id | name
-------+--------------+---------
     1 |            1 | Joe
     1 |            2 | Ed
     2 |            1 | Mary
     1 |            3 | Mike
```

```
(4 rows)
```

最後，交易會遞交或轉返。

```
COMMIT;
```

預存程序傳回的游標受制於 DECLARE CURSOR 中所述的相同限制和效能考量。如需詳細資訊，請參閱游標限制條件。

下列範例顯示從 JDBC 使用 refcursor 資料類型呼叫 get_result_set 預存程序。常值 'mycursor' (游標名稱) 會傳給 prepareStatement。然後，從 ResultSet 擷取結果。

```
static void refcursor_example(Connection conn) throws SQLException {
    conn.setAutoCommit(false);
    PreparedStatement proc = conn.prepareStatement("CALL get_result_set(1,
 'mycursor')");
    proc.execute();
    ResultSet rs = statement.executeQuery("fetch all from mycursor");
    while (rs.next()) {
      int n = rs.getInt(1);
      System.out.println("n " + n);
    }
```

使用暫存資料表

若要傳回結果，您可以傳回含有結果列之暫存資料表的控制代碼。用戶端可以將名稱當作參數提供給預存程序。在預存程序內，可使用動態 SQL 來操作暫存資料表。下列顯示一個範例。

```
CREATE PROCEDURE get_result_set(param IN integer, tmp_name INOUT varchar(256)) as $$
DECLARE
  row record;
BEGIN
  EXECUTE 'drop table if exists ' || tmp_name;
  EXECUTE 'create temp table ' || tmp_name || ' as select * from fact_tbl where id <= '
 || param;
END;
$$ LANGUAGE plpgsql;

CALL get_result_set(2, 'myresult');
 tmp_name
-----------
 myresult
```

```
(1 row)

SELECT * from myresult;
 id | secondary_id | name
----+--------------+------
  1 |            1 | Joe
  2 |            1 | Mary
  1 |            2 | Ed
  1 |            3 | Mike
(4 rows)
```

管理交易

您可以建立具有預設交易管理行為或非原子行為的預存程序。

預設模式預存程序交易管理

預設交易模式自動遞交行為會導致系統個別遞交每個分開執行的 SQL 命令。系統會將對預存程序的呼叫視為單一 SQL 命令。程序內的 SQL 陳述式就像在交易區塊內執行一樣，此區塊是在呼叫開始時隱含地開始，而在呼叫完成時結果。對另一個程序的巢狀呼叫就如同任何其他 SQL 陳述式，在發起人的相同交易範圍內運作。如需自動遞交行為的詳細資訊，請參閱可序列化隔離。

但是，假設您從使用者指定的交易區塊 (由 BEGIN...COMMIT 定義) 中呼叫預存程序。在此情況下，預存程序中的所有陳述式都會在使用者指定交易的內容中執行。程序不會隱含地在結束時遞交。發起人控制程序遞交或轉返。

如果執行預存程序時發生任何錯誤，您在目前交易中做的所有變更都會復原。

您可以使用下列交易來控制預存程序中的陳述式：

- COMMIT – 遞交目前交易中完成的所有工作，並隱含地啟動新交易。如需詳細資訊，請參閱COMMIT。
- ROLLBACK – 回復目前交易中完成的工作，並隱含地啟動新交易。如需詳細資訊，請參閱ROLLBACK。

TRUNCATE 是另一個可以在預存程序內發出，且會影響交易管理作業的的陳述式。在 Amazon Redshift 中，TRUNCATE 會隱含地發出遞交。在預存程序的範圍內，此行為不變。從預存程序內發出 TRUNCATE 陳述式時，它會遞交目前的交易並啟動新交易。如需詳細資訊，請參閱TRUNCATE。

COMMIT、ROLLBACK 或 TRUNCATE 陳述式後面的所有陳述式都會在新交易的範圍內執行，一直到遇到另一個 COMMIT、ROLLBACK 或 TRUNCATE 陳述式或預存程序結束為止。

當您從預存程序內使用 COMMIT、ROLLBACK 或 TRUNCATE 陳述式時，有下列限制：

- 如果從交易區塊內呼叫預存程序，就無法發出 COMMIT、ROLLBACK 或 TRUNCATE 陳述式。這項限制適用於預存程序本身和任何巢狀程序呼叫內部。
- 如果使用 SET config 選項建立預存程序，就無法發出 COMMIT、ROLLBACK 或 TRUNCATE 陳述式。這項限制適用於預存程序本身和任何巢狀程序呼叫內部。
- 處理 COMMIT、ROLLBACK 或 TRUNCATE 陳述式時，任何已開啟的 (明確或隱含) 游標皆會自動關閉。如需了解明確和隱含游標的限制，請參閱預存程序支援的考量。

此外，您不能使用動態 SQL 執行 COMMIT 或 ROLLBACK，但您可以使用動態 SQL 執行 TRUNCATE。如需詳細資訊，請參閱動態 SQL。

當您使用預存程序時，需考慮到 PL/pgSQL 中的 BEGIN 和 END 陳述式僅可用來分組，而無法用來啟動或結束交易。如需詳細資訊，請參閱區塊。

下列範例示範從明確交易區塊內呼叫預存程序時的交易行為。從預存程序外發出的兩個 insert 陳述式和從它之內發出的一個 insert 陳述式，都是相同交易 (3382) 的一部分。當使用者發出明確遞交時，交易就會遞交。

```
CREATE OR REPLACE PROCEDURE sp_insert_table_a(a int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table_a values (a);
END;
$$;

Begin;
  insert into test_table_a values (1);
  Call sp_insert_table_a(2);
  insert into test_table_a values (3);
Commit;

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  | pid |  type   |               stmt_text
--------+------+-----+---------+----------------------------------------
    103 | 3382 | 599 | UTILITY | Begin;
    103 | 3382 | 599 | QUERY   | insert into test_table_a values (1);
    103 | 3382 | 599 | UTILITY | Call sp_insert_table_a(2);
```

```
    103 | 3382 | 599 | QUERY   | INSERT INTO test_table_a values ( $1 )
    103 | 3382 | 599 | QUERY   | insert into test_table_a values (3);
    103 | 3382 | 599 | UTILITY | COMMIT
```

相對的，假設有同樣的陳述式從明確交易區塊外發出，而且該工作階段的自動遞交設定為 ON。在這種情況下，每個陳述式都會在自己的交易內運作。

```
insert into test_table_a values (1);
Call sp_insert_table_a(2);
insert into test_table_a values (3);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  | pid |  type   |
             stmt_text
--------+------+-----+---------
+-----------------------------------------------------------------------------------------------------
    103 | 3388 | 599 | QUERY   | insert into test_table_a values (1);
    103 | 3388 | 599 | UTILITY | COMMIT
    103 | 3389 | 599 | UTILITY | Call sp_insert_table_a(2);
    103 | 3389 | 599 | QUERY   | INSERT INTO test_table_a values ( $1 )
    103 | 3389 | 599 | UTILITY | COMMIT
    103 | 3390 | 599 | QUERY   | insert into test_table_a values (3);
    103 | 3390 | 599 | UTILITY | COMMIT
```

下列範例會在插入 test_table_a 後發出 TRUNCATE 陳述式。TRUNCATE 陳述式發出隱含遞交，以遞交目前交易 (3335) 並啟動新的交易 (3336)。新的交易會在程序結束時遞交。

```
CREATE OR REPLACE PROCEDURE sp_truncate_proc(a int, b int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table_a values (a);
  TRUNCATE test_table_b;
  INSERT INTO test_table_b values (b);
END;
$$;

Call sp_truncate_proc(1,2);

select userid, xid, pid, type, trim(text) as stmt_text
```

```
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |  pid  |  type   |
                                        stmt_text
--------+------+-------+---------
+-----------------------------------------------------------------------------------------------------
    103 | 3335 | 23636 | UTILITY | Call sp_truncate_proc(1,2);
    103 | 3335 | 23636 | QUERY   | INSERT INTO test_table_a values ( $1 )
    103 | 3335 | 23636 | UTILITY | TRUNCATE test_table_b
    103 | 3335 | 23636 | UTILITY | COMMIT
    103 | 3336 | 23636 | QUERY   | INSERT INTO test_table_b values ( $1 )
    103 | 3336 | 23636 | UTILITY | COMMIT
```

下列範例從巢狀呼叫發出 TRUNCATE。TRUNCATE 遞交目前為止在交易 (3344) 的外層和內層程序中完成的所有工作。它會啟動新的交易 (3345)。新的交易會在外層程序結束時遞交。

```
CREATE OR REPLACE PROCEDURE sp_inner(c int, d int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO inner_table values (c);
  TRUNCATE outer_table;
  INSERT INTO inner_table values (d);
END;
$$;

CREATE OR REPLACE PROCEDURE sp_outer(a int, b int, c int, d int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO outer_table values (a);
  Call sp_inner(c, d);
  INSERT INTO outer_table values (b);
END;
$$;

Call sp_outer(1, 2, 3, 4);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |  pid  |  type   |
                                        stmt_text
```

```
--------+------+-------+---------
+-----------------------------------------------------------------------------------------------
    103 | 3344 | 23636 | UTILITY | Call sp_outer(1, 2, 3, 4);
    103 | 3344 | 23636 | QUERY   | INSERT INTO outer_table values ( $1 )
    103 | 3344 | 23636 | UTILITY | CALL sp_inner( $1 , $2 )
    103 | 3344 | 23636 | QUERY   | INSERT INTO inner_table values ( $1 )
    103 | 3344 | 23636 | UTILITY | TRUNCATE outer_table
    103 | 3344 | 23636 | UTILITY | COMMIT
    103 | 3345 | 23636 | QUERY   | INSERT INTO inner_table values ( $1 )
    103 | 3345 | 23636 | QUERY   | INSERT INTO outer_table values ( $1 )
    103 | 3345 | 23636 | UTILITY | COMMIT
```

下列範例顯示游標 cur1 已在 TRUNCATE 陳述式遞交時關閉。

```
CREATE OR REPLACE PROCEDURE sp_open_cursor_truncate()
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  cur1 cursor for select * from test_table_a order by 1;
BEGIN
  open cur1;
  TRUNCATE table test_table_b;
  Loop
    fetch cur1 into rec;
    raise info '%', rec.c1;
    exit when not found;
  End Loop;
END
$$;

call sp_open_cursor_truncate();
ERROR: cursor "cur1" does not exist
CONTEXT: PL/pgSQL function "sp_open_cursor_truncate" line 8 at fetch
```

下列範例會發出 TRUNCATE 陳述式，但不能從明確交易區塊內進行呼叫。

```
CREATE OR REPLACE PROCEDURE sp_truncate_atomic() LANGUAGE plpgsql
AS $$
BEGIN
  TRUNCATE test_table_b;
END;
$$;
```

```
Begin;
  Call sp_truncate_atomic();
ERROR: TRUNCATE cannot be invoked from a procedure that is executing in an atomic
 context.
HINT: Try calling the procedure as a top-level call i.e. not from within an explicit
 transaction block.
Or, if this procedure (or one of its ancestors in the call chain) was created with SET
 config options, recreate the procedure without them.
CONTEXT: SQL statement "TRUNCATE test_table_b"
PL/pgSQL function "sp_truncate_atomic" line 2 at SQL statement
```

下列範例顯示不是超級使用者或資料表擁有者的使用者可以對資料表發出 TRUNCATE 陳述式。使用者會使用 Security Definer 預存程序來執行此動作。該範例會出現下列操作：

- user1 建立表格 test_tbl。
- user1 建立預存程序 sp_truncate_test_tbl。
- user1 授予預存程序的 EXECUTE 權限給 user2。
- user2 執行預存程序以截斷表格 test_tbl。此範例顯示 TRUNCATE 命令前後的資料列計數。

```
set session_authorization to user1;
create table test_tbl(id int, name varchar(20));
insert into test_tbl values (1,'john'), (2, 'mary');
CREATE OR REPLACE PROCEDURE sp_truncate_test_tbl() LANGUAGE plpgsql
AS $$
DECLARE
  tbl_rows int;
BEGIN
  select count(*) into tbl_rows from test_tbl;
  RAISE INFO 'RowCount before Truncate: %', tbl_rows;
  TRUNCATE test_tbl;
  select count(*) into tbl_rows from test_tbl;
  RAISE INFO 'RowCount after Truncate: %', tbl_rows;
END;
$$ SECURITY DEFINER;
grant execute on procedure sp_truncate_test_tbl() to user2;
reset session_authorization;


set session_authorization to user2;
call sp_truncate_test_tbl();
```

```
INFO:  RowCount before Truncate: 2
INFO:  RowCount after Truncate: 0
CALL
reset session_authorization;
```

下列範例會發出 COMMIT 兩次。第一個 COMMIT 會遞交 10363 交易中完成的所有工作，並隱含地啟動 10364 交易。第二個 COMMIT 陳述式則會遞交 10364 交易。

```
CREATE OR REPLACE PROCEDURE sp_commit(a int, b int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table values (a);
  COMMIT;
  INSERT INTO test_table values (b);
  COMMIT;
END;
$$;

call sp_commit(1,2);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;
 userid |  xid  | pid  |  type   |
                               stmt_text
--------+-------+------+---------
+-----------------------------------------------------------------------------------------------------------------
    100 | 10363 | 3089 | UTILITY | call sp_commit(1,2);
    100 | 10363 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
    100 | 10363 | 3089 | UTILITY | COMMIT
    100 | 10364 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
    100 | 10364 | 3089 | UTILITY | COMMIT
```

在 sum_vals 大於 2 的情況下，下列範例會發出 ROLLBACK 陳述式。第一個 ROLLBACK 陳述式會回復 10377 交易中完成的所有工作，並啟動新的 10378 交易；10378 交易會在程序結束時遞交。

```
CREATE OR REPLACE PROCEDURE sp_rollback(a int, b int) LANGUAGE plpgsql
AS $$
DECLARE
  sum_vals int;
BEGIN
  INSERT INTO test_table values (a);
  SELECT sum(c1) into sum_vals from test_table;
```

```
  IF sum_vals > 2 THEN
    ROLLBACK;
  END IF;

  INSERT INTO test_table values (b);
END;
$$;

call sp_rollback(1, 2);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

userid |  xid  | pid  |  type   |
                               stmt_text
--------+-------+------+---------
+-----------------------------------------------------------------------------------------------------------
    100 | 10377 | 3089 | UTILITY | call sp_rollback(1, 2);
    100 | 10377 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
    100 | 10377 | 3089 | QUERY   | SELECT sum(c1) from test_table
    100 | 10377 | 3089 | QUERY   | Undoing 1 transactions on table 133646 with current
 xid 10377 : 10377
    100 | 10378 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
    100 | 10378 | 3089 | UTILITY | COMMIT
```

非原子模式預存程序交易管理

在 NONATIC 模式中建立的預存程序與以預設模式建立的程序有不同的交易控制行為。與預存程序外部 SQL 命令的自動遞交行為類似，NONATOMIC 程序內的每個 SQL 陳述式都會在自己的交易中執行，並自動遞交。如果使用者在 NONATOMIC 預存程序中開始明確交易區塊，則區塊內的 SQL 陳述式不會自動確認。交易區塊控制項遞交或復原其中的陳述式。

在 NONATOMIC 預存程序中，您可以使用 START TRANSACTION 陳述式在程序內開啟明確的交易區塊。不過，如果已經有開啟的交易區塊，這個陳述式就不會執行任何動作，因為 Amazon Redshift 不支援子交易。上一筆交易會繼續進行。

當您在 NONATOMIC 程序內使用游標 FOR 迴圈時，請確定您在迭代查詢結果之前開啟明確的交易區塊。否則，當迴圈內的 SQL 陳述式自動遞交時，游標會關閉。

使用 NONATOMIC 模式行為時的一些注意事項如下：

- 如果沒有開啟的交易區塊，且工作階段已自動遞交預存程序中的每個 SQL 陳述式設定為 ON。

- 如果從交易區塊內呼叫預存程序，您可以發出 COMMIT/ROLLBACK/TRUNCATE 陳述式來結束交易。這在預設模式下是不可能的。
- 您可以發出 START TRANSACTION 陳述式，以便在預存程序中開始交易區塊。

下列範例會示範使用 NONATOMIC 預存程序時的交易行為。下列所有範例的工作階段都將自動遞交設定為 ON。

在下列範例中，NONATOMIC 預存程序會有兩個 INSERT 陳述式。在交易區塊外呼叫程序時，程序中的每個 INSERT 陳述式都會自動遞交。

```
CREATE TABLE test_table_a(v int);
CREATE TABLE test_table_b(v int);

CREATE OR REPLACE PROCEDURE sp_nonatomic_insert_table_a(a int, b int) NONATOMIC AS
$$
BEGIN
    INSERT INTO test_table_a values (a);
    INSERT INTO test_table_b values (b);
END;
$$
LANGUAGE plpgsql;

Call sp_nonatomic_insert_table_a(1,2);

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |    pid     |  type   |               stmt_text
--------+------+------------+---------+----------------------------------------
      1 | 1792 | 1073807554 | UTILITY | Call sp_nonatomic_insert_table_a(1,2);
      1 | 1792 | 1073807554 | QUERY   | INSERT INTO test_table_a values ( $1 )
      1 | 1792 | 1073807554 | UTILITY | COMMIT
      1 | 1793 | 1073807554 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1793 | 1073807554 | UTILITY | COMMIT
(5 rows)
```

但是，當從 BEGIN..COMMIT 區塊內呼叫程序時，所有陳述式都是相同交易的一部分 (xid=1799)。

```
Begin;
  INSERT INTO test_table_a values (10);
```

```
  Call sp_nonatomic_insert_table_a(20,30);
  INSERT INTO test_table_b values (40);
Commit;

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |    pid     |  type   |                stmt_text
--------+------+------------+---------+-----------------------------------------
      1 | 1799 | 1073914035 | UTILITY | Begin;
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_a values (10);
      1 | 1799 | 1073914035 | UTILITY | Call sp_nonatomic_insert_table_a(20,30);
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_a values ( $1 )
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1799 | 1073914035 | QUERY   | INSERT INTO test_table_b values (40);
      1 | 1799 | 1073914035 | UTILITY | COMMIT
(7 rows)
```

在這個範例中，兩個 INSERT 陳述式在 START TRANSACTION...COMMIT 之間。當程序在交易區塊之外呼叫時，兩個 INSERT 陳述式位於相同的交易中 (xid=1866)。

```
CREATE OR REPLACE PROCEDURE sp_nonatomic_txn_block(a int, b int) NONATOMIC AS
$$
BEGIN
    START TRANSACTION;
    INSERT INTO test_table_a values (a);
    INSERT INTO test_table_b values (b);
    COMMIT;
END;
$$
LANGUAGE plpgsql;

Call sp_nonatomic_txn_block(1,2);

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |    pid     |  type   |               stmt_text
--------+------+------------+---------+----------------------------------------
      1 | 1865 | 1073823998 | UTILITY | Call sp_nonatomic_txn_block(1,2);
      1 | 1866 | 1073823998 | QUERY   | INSERT INTO test_table_a values ( $1 )
```

```
      1 | 1866 | 1073823998 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1866 | 1073823998 | UTILITY | COMMIT
(4 rows)
```

當從 BEGIN...COMMIT 區塊內呼叫程序時，程序內的 START 交易不會執行任何動作，因為已經有開啟的交易。程序中的 COMMIT 會遞交目前交易 (xid = 1876) 並啟動一個新交易。

```
Begin;
  INSERT INTO test_table_a values (10);
  Call sp_nonatomic_txn_block(20,30);
  INSERT INTO test_table_b values (40);
Commit;

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
 sequence;

 userid | xid  |    pid     |  type   |               stmt_text
--------+------+------------+---------+----------------------------------------
      1 | 1876 | 1073832133 | UTILITY | Begin;
      1 | 1876 | 1073832133 | QUERY   | INSERT INTO test_table_a values (10);
      1 | 1876 | 1073832133 | UTILITY | Call sp_nonatomic_txn_block(20,30);
      1 | 1876 | 1073832133 | QUERY   | INSERT INTO test_table_a values ( $1 )
      1 | 1876 | 1073832133 | QUERY   | INSERT INTO test_table_b values ( $1 )
      1 | 1876 | 1073832133 | UTILITY | COMMIT
      1 | 1878 | 1073832133 | QUERY   | INSERT INTO test_table_b values (40);
      1 | 1878 | 1073832133 | UTILITY | COMMIT
(8 rows)
```

此範例示範如何使用游標迴圈。test_table_a 資料表具有三個值。我們的目標是迭代這三個值，並將它們插入到資料表 test_table_b。如果 NONATOMIC 預存程序程以下面的方式建立，在第一個迴圈中執行 INSERT 陳述式後，它將擲出錯誤游標「cur1」不存在。這是因為 INSERT 的自動遞交會關閉開啟的游標。

```
insert into test_table_a values (1), (2), (3);

CREATE OR REPLACE PROCEDURE sp_nonatomic_cursor() NONATOMIC
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  cur1 cursor for select * from test_table_a order by 1;
```

```
BEGIN
  open cur1;
  Loop
    fetch cur1 into rec;
    exit when not found;
    raise info '%', rec.v;
    insert into test_table_b values (rec.v);
  End Loop;
END
$$;

CALL sp_nonatomic_cursor();

INFO:  1
ERROR:  cursor "cur1" does not exist
CONTEXT:  PL/pgSQL function "sp_nonatomic_cursor" line 7 at fetch
```

若要讓游標迴圈運作，請將其放在 START TRANSACTION...COMMIT 之間。

```
insert into test_table_a values (1), (2), (3);

CREATE OR REPLACE PROCEDURE sp_nonatomic_cursor() NONATOMIC
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  cur1 cursor for select * from test_table_a order by 1;
BEGIN
  START TRANSACTION;
  open cur1;
  Loop
    fetch cur1 into rec;
    exit when not found;
    raise info '%', rec.v;
    insert into test_table_b values (rec.v);
  End Loop;
  COMMIT;
END
$$;

CALL sp_nonatomic_cursor();

INFO:  1
```

```
INFO:  2
INFO:  3
CALL
```

捕捉錯誤

當預存程序中的查詢或命令造成錯誤時，後續查詢不會執行，而且會復原交易。但您可以使用 EXCEPTION 區塊來處理錯誤。

> **Note**
>
> 預設行為是，即使預存程序中沒有其他產生錯誤的條件，錯誤也會導致後續查詢無法執行。

```
[ <<label>> ]
[ DECLARE
  declarations ]
BEGIN
  statements
EXCEPTION
  WHEN OTHERS THEN
    statements
END;
```

當發生例外狀況，而您新增例外狀況處理區塊時，您可以撰寫 RAISE 陳述式和大多數其他 PL/pgSQL 陳述式。例如，您可以使用自訂訊息引發例外狀況，或將記錄插入記錄資料表。

進入例外狀況處理區塊時，會復原目前的交易，並建立新的交易以執行區塊中的陳述式。如果區塊中的陳述式沒有錯誤地執行，就會遞交交易並重新擲回例外狀況。最後，預存程序會結束。

例外狀況區塊中唯一支援的條件是 OTHERS，可比對各種錯誤類型 (查詢取消除外)。此外，如果例外狀況處理塊中發生錯誤，則外部例外狀況處理塊可以捕獲該錯誤。

當 NONATOMIC 程序內部發生錯誤時，如果錯誤是由例外狀況區塊處理，則不會重新擲回錯誤。請參閱 PL/pgSQL 陳述式 RAISE 以擲出例外狀況處理區塊捕獲的例外狀況。此陳述式僅在例外狀況處理區塊中有效。如需更多資訊，請參閱 RAISE。

使用 CONTINUE 處理常式控制預存程序中發生錯誤之後的情況

CONTINUE 處理常式是一種例外狀況處理常式，可控制 NONATOMIC 預存程序內的執行流程。透過使用的過程，您可以揪出問題點並處理異常，而不需結束現有陳述式區塊。通常當預存程序中發生錯誤

時，流程會中斷，並將錯誤傳回給呼叫者。但是，在某些使用案例中，錯誤情況還沒有嚴重到足以中斷流程。您可能想要按既定程序處理錯誤，使用您在不同的交易中選擇的錯誤處理邏輯，然後繼續執行錯誤後續的陳述式。語法如下列所示。

```
[ DECLARE
  declarations ]
BEGIN
  statements
EXCEPTION
  [ CONTINUE_HANDLER | EXIT_HANDLER ] WHEN OTHERS THEN
    handler_statements
END;
```

有數個系統資料表可協助您收集有關各種錯誤類型的資訊。如需詳細資訊，請參閱 STL_LOAD_ERRORS、STL_ERROR 及 SYS_STREAM_SCAN_ERRORS。您也可以使用其他系統資料表來疑難排解錯誤。有關這些的更多資訊，請參閱系統資料表和檢視參考。

範例

下列範例說明如何在例外狀況處理區塊中撰寫陳述式。預存程序正在使用預設的交易管理行為。

```
CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);

CREATE OR REPLACE PROCEDURE update_employee_sp() AS
$$
BEGIN
    UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
    EXECUTE 'select invalid';
EXCEPTION WHEN OTHERS THEN
    RAISE INFO 'An exception occurred.';
    INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp();

INFO:  An exception occurred.
ERROR:  column "invalid" does not exist
CONTEXT:  SQL statement "select invalid"
```

```
PL/pgSQL function "update_employee_sp" line 3 at execute statement
```

在這個範例中，如果我們呼叫 update_employee_sp，會引發資訊訊息「發生例外狀況。」並將錯誤訊息插入記錄資料表的 employee_error_log 日誌中。在預存程序結束之前，會再次擲回原始例外狀況。下列查詢顯示執行範例所產生的記錄。

```
SELECT * from employee;

firstname | lastname
-----------+----------
 Tomas     | Smith

SELECT * from employee_error_log;

          message
-----------------------------------------------
 Error message: column "invalid" does not exist
```

如需 RAISE 的詳細資訊，包括格式化說明和其他層級清單，請參閱支援的 PL/pgSQL 陳述式。

下列範例說明如何在例外狀況處理區塊中撰寫陳述式。預存程序正在使用 NONATOMIC 交易管理行為。在此範例中，程序呼叫完成後，不會擲回呼叫者的錯誤。UPDATE 陳述式不會因為下一個陳述式中的錯誤而復原。會引發資訊訊息，並在記錄資料表中插入錯誤訊息。

```
CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);

-- Create the SP in NONATOMIC mode
CREATE OR REPLACE PROCEDURE update_employee_sp_2() NONATOMIC AS
$$
BEGIN
    UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
    EXECUTE 'select invalid';
EXCEPTION WHEN OTHERS THEN
    RAISE INFO 'An exception occurred.';
    INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp_2();
```

```
INFO:  An exception occurred.
CALL

SELECT * from employee;

 firstname | lastname
-----------+----------
 Adam      | Smith
(1 row)

SELECT * from employee_error_log;

                    message
-----------------------------------------------
 Error message: column "invalid" does not exist
(1 row)
```

此範例示範如何使用兩個子區塊建立程序。呼叫預存程序時，第一個子區塊的錯誤會由其例外狀況處理區塊處理。在第一個子區塊完成之後，程序會繼續執行第二個子區塊。從結果可以看到，程序呼叫完成時沒有擲回任何錯誤。資料表 employee 上的 UPDATE 和 INSERT 操作已遞交。來自兩個例外狀況區塊的錯誤訊息都會插入記錄資料表中。

```
CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);

CREATE OR REPLACE PROCEDURE update_employee_sp_3() NONATOMIC AS
$$
BEGIN
    BEGIN
        UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
        EXECUTE 'select invalid1';
    EXCEPTION WHEN OTHERS THEN
        RAISE INFO 'An exception occurred in the first block.';
        INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
    END;
    BEGIN
        INSERT INTO employee VALUES ('Edie','Robertson');
        EXECUTE 'select invalid2';
    EXCEPTION WHEN OTHERS THEN
        RAISE INFO 'An exception occurred in the second block.';
        INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
    END;
```

```
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp_3();
INFO:  An exception occurred in the first block.
INFO:  An exception occurred in the second block.
CALL

SELECT * from employee;

 firstname | lastname
-----------+-----------
 Adam      | Smith
 Edie      | Robertson
(2 rows)

SELECT * from employee_error_log;

                     message
-------------------------------------------------
 Error message: column "invalid1" does not exist
 Error message: column "invalid2" does not exist
(2 rows)
```

下列範例示範如何使用 CONTINUE 例外狀況處理常式。此範例會建立兩份資料表，並在預存程序中使用資料表。CONTINUE 處理常式會以 NONATOMIC 交易管理行為控制預存程序中的執行流程。

```
CREATE TABLE tbl_1 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_1() NONATOMIC AS
$$
BEGIN
    INSERT INTO tbl_1 VALUES (1);
    -- Expect an error for the insert statement following, because of the invalid value
    INSERT INTO tbl_1 VALUES ("val");
    INSERT INTO tbl_1 VALUES (2);
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

呼叫預存程序：

```
CALL sp_exc_handling_1();
```

流程如下所示：

1. 因為嘗試在資料欄插入不相容的資料類型而發生錯誤。控制傳遞給 EXCEPTION 區塊。當進入例外狀況處理區塊時，會復原目前的交易，並建立新的隱含交易以執行區塊中的陳述式。
2. 如果 CONTINUE_HANDLER 中的陳述式執行無錯誤，則控制權會傳遞至立即接續在導致例外狀況的陳述式之後的陳述式。(如果 CONTINUE_HANDLER 中的陳述式引發新的異常，則可以使用 EXCEPTION 區塊中的處理程序來處理它。)

呼叫範例預存程序之後，資料表包含下列記錄：

- 如果執行 SELECT * FROM tbl_1;，它會傳回兩個記錄。這些包含值 1 和 2。
- 如果執行 SELECT * FROM tbl_error_logging;，它會傳回一個包含下列值的記錄：發生的錯誤、42703 和 tbl_1 中不存在資料欄 "val"。

下列其他錯誤處理範例會同時使用 EXIT 處理常式和 CONTINUE 處理常式。此範例建立兩個表：一個資料表和一個記錄表。另外還建立一個展示如何處理錯誤的預存程序：

```
CREATE TABLE tbl_1 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_2() NONATOMIC AS
$$
BEGIN
    INSERT INTO tbl_1 VALUES (1);
    BEGIN
        INSERT INTO tbl_1 VALUES (100);
        -- Expect an error for the insert statement following, because of the invalid
 value
        INSERT INTO tbl_1 VALUES ("val");
        INSERT INTO tbl_1 VALUES (101);
    EXCEPTION EXIT_HANDLER WHEN OTHERS THEN
        INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
    END;
    INSERT INTO tbl_1 VALUES (2);
    -- Expect an error for the insert statement following, because of the invalid value
    INSERT INTO tbl_1 VALUES ("val");
```

```
    INSERT INTO tbl_1 VALUES (3);
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

建立預存程序後，使用下列命令呼叫：

```
CALL sp_exc_handling_2();
```

當內部例外區塊 (由 BEGIN 和 END 的內部集合括住)中發生錯誤時，會由 EXIT 處理常式處理。外部區塊中發生的任何錯誤則由 CONTINUE 處理常式處理。

呼叫範例預存程序之後，資料表包含下列記錄：

- 如果執行 SELECT * FROM tbl_1;，它會傳回四個記錄，其值為 1、2、3 和 100。
- 如果執行 SELECT * FROM tbl_error_logging;，它會傳回兩個記錄。它們具有以下值：發生的錯誤、42703 和 tbl_1 中不存在資料欄 "val"。

如果表 tbl_error_logging 不存在，則會引發例外狀況。

下列範例展示如何使用 CONTINUE 例外狀況處理常式與 FOR 迴圈。此範例會建立三個資料表，並在預存程序內的 FOR 迴圈中使用這些資料表。FOR 迴圈是結果集變體，這要示它會反覆運算查詢結果：

```
CREATE TABLE tbl_1 (a int);
INSERT INTO tbl_1 VALUES (1), (2), (3);
CREATE TABLE tbl_2 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_loop() NONATOMIC AS
$$
DECLARE
 rec RECORD;
BEGIN
    FOR rec IN SELECT a FROM tbl_1
    LOOP
        IF rec.a = 2 THEN
            -- Expect an error for the insert statement following, because of the
 invalid value
            INSERT INTO tbl_2 VALUES("val");
```

```
        ELSE
            INSERT INTO tbl_2 VALUES (rec.a);
        END IF;
    END LOOP;
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

呼叫預存程序：

```
CALL sp_exc_handling_loop();
```

呼叫範例預存程序之後，資料表包含下列記錄：

- 如果執行 `SELECT * FROM tbl_2;`，它會傳回兩個記錄。這些包含值 1 和 3。
- 如果執行 `SELECT * FROM tbl_error_logging;`，它會傳回一個包含下列值的記錄：發生的錯誤、42703 和 tbl_2 中不存在資料欄 "val"。

有關 CONTINUE 處理常式的使用須知：

- CONTINUE_HANDLER 和 EXIT_HANDLER 關鍵字只能在 NONATOMIC 預存程序中使用。
- CONTINUE_HANDLER 和 EXIT_HANDLER 關鍵字是選用的。EXIT_HANDLER 是預設值。

記錄預存程序日誌

預存程序的詳細資訊都記錄在下列系統資料表和檢視中：

- SVL_STORED_PROC_CALL – 記錄預存程序呼叫的開始時間和結束時間，以及呼叫是否在完成之前結束等詳細資訊。如需詳細資訊，請參閱SVL_STORED_PROC_CALL。
- SVL_STORED_PROC_MESSAGES – RAISE 查詢發出之預存程序中的訊息，將以對應的記錄層級記錄。如需詳細資訊，請參閱SVL_STORED_PROC_MESSAGES。
- SVL_QLOG – 針對從預存程序呼叫的每個查詢，記錄程序呼叫的查詢 ID。如需詳細資訊，請參閱SVL_QLOG。
- STL_UTILITYTEXT – 在預存程序呼叫完成之後記錄。如需詳細資訊，請參閱STL_UTILITYTEXT。
- PG_PROC_INFO – 此系統目錄檢視顯示預存程序的相關資訊。如需詳細資訊，請參閱PG_PROC_INFO。

預存程序支援的考量

使用 Amazon Redshift 預存程序時有下列限制。

Amazon Redshift 和 PostgreSQL 在預存程序支援方面的差異

以下為 Amazon Redshift 和 PostgreSQL 中的預存程序支援的差異：

- Amazon Redshift 不支援子交易，因此對例外狀況處理區塊的支援有限。

考量與限制

以下是 Amazon Redshift 中預存程序的考量事項：

- 資料庫的預存程序數目上限為 10,000。
- 程序的原始程式碼大小上限為 2 MB。
- 您在使用者工作階段中可同時開啟的明確和隱含游標數目上限是一個。在 SQL 陳述式的結果集上反覆運算的 FOR 迴圈會開啟隱含游標。不支援巢狀游標。
- 明確和隱含游標的結果集大小限制，與標準 Amazon Redshift 游標相同。如需詳細資訊，請參閱游標限制條件。
- 巢狀呼叫層數上限為 16。
- 輸入引數的程序參數數目上限為 32，輸出引數的上限也是 32。
- 預存程序中的變數數目上限為 1,024。
- 預存程序內不支援需要有自己的交易範圍的任何 SQL 命令。範例包括：
  - PREPARE
  - CREATE/DROP DATABASE
  - CREATE EXTERNAL TABLE
  - VACUUM
  - SET LOCAL
  - ALTER TABLE APPEND
- 對於 `refcursor` 資料類型，不支援透過 Java 資料庫連線 (JDBC) 驅動程式發出的 `registerOutParameter` 方法呼叫。關於使用 `refcursor` 資料類型的範例，請參閱傳回結果集。

PL/pgSQL 語言參考

Amazon Redshift 中的預存程序以 PostgreSQL PL/pgSQL 程序性語言為基礎，但有一些重要差異。在此參考中，您可以找到 Amazon Redshift 所實作的 PL/pgSQL 語法的詳細資訊。如需 PL/pgSQL 的相關資訊，請參閱 PostgreSQL 文件中的 PL/pgSQL - SQL 程序性語言。

主題

PL/pgSQL 參考慣例

在本節中，您可以找到 PL/pgSQL 預存程序語言的語法編寫慣例。

字元	描述
CAPS (大寫字母)	大寫字詞為關鍵字。
[ ]	方括號是用來表示選用的引數。方括號中的多個引數，代表您可以選擇任何數目的引數。此外，方括號中不同行的引數，代表 Amazon Redshift 剖析器期望引數的排序，等於在語法中所列出的順序。
{ }	大括號表示您需要選擇大括號內的其中一個引數。
|	直立線符號會將您可選擇的引數隔開。
####	紅色斜體的字表示預留位置。插入適當的值來取代紅色斜體字詞。
. . .	省略符號表示您可以重複前一個元素。
'	單引號中的字詞表示您必須輸入引用內容。

PL/pgSQL 的結構

PL/pgSQL 是一種程序性語言，具有很多建構與其他程序性語言相同。

主題

區塊

PL/pgSQL 是區塊結構化語言。程序的整個主體都定義在區塊內，其中包含變數宣告和 PL/pgSQL 陳述式。陳述式也可以是巢狀區塊或子區塊。

以分號作為宣告和陳述式的結尾。在區塊或子區塊中的 END 關鍵字後面加上分號。請勿在關鍵字 DECLARE 和 BEGIN 後面使用分號。

您可以混合大小寫來撰寫所有關鍵字和識別碼。識別碼除非以雙引號括住，否則會隱含轉換為小寫。

雙連字號 (--) 表示註解開始，一直延續到行尾。/* 表示區塊註解開始，一直延續到出現下一個 */。您不能將區塊註解巢狀化。不過，您可以用區塊註解圍住雙連字號註解，雙連字號可以隱藏區塊註解分隔符號 /* 和 */。

在區塊的陳述式區段中，任何陳述式可以是子區塊。您可以使用子區塊進行邏輯分組，或將變數區域化給一小組陳述式。

```
[ <<label>> ]
[ DECLARE
  declarations ]
BEGIN
  statements
END [ label ];
```

在區塊前面的宣告區段中所宣告的變數，在每次進入區塊時會初始化為其預設值。換言之，在每次函數呼叫時不只初始化一次。

下列顯示一個範例。

```
CREATE PROCEDURE update_value() AS $$
DECLARE
  value integer := 20;
BEGIN
```

```
  RAISE NOTICE 'Value here is %', value;  -- Value here is 20
  value := 50;
  --
  -- Create a subblock
  --
  DECLARE
    value integer := 80;
  BEGIN
    RAISE NOTICE 'Value here is %', value;  -- Value here is 80
  END;

  RAISE NOTICE 'Value here is %', value;  -- Value here is 50
END;
$$ LANGUAGE plpgsql;
```

使用標籤來識別要在 EXIT 陳述式中使用的區塊，或限定區塊中宣告的變數名稱。

請勿混淆 PL/pgSQL 中用於陳述式分組的 BEGIN/END 與用於交易控制的資料庫命令。PL/pgSQL 中的 BEGIN 和 END 只用於分組。而無法用來啟動或結束交易。

變數宣告

在區塊的 DECLARE 區段中宣告區塊的所有變數，但迴圈變數除外。變數可以使用任何有效的 Amazon Redshift 資料類型。關於支援的資料類型，請參閱資料類型。

PL/pgSQL 變數可以是 Amazon Redshift 支援的任何資料類型，還有 RECORD 和 `refcursor`。如需 RECORD 的相關資訊，請參閱 記錄類型。如需 `refcursor` 的相關資訊，請參閱 游標。

```
DECLARE
name [ CONSTANT ] type [ NOT NULL ] [ { DEFAULT | := } expression ];
```

以下，您可以找到變數宣告範例。

```
customerID integer;
numberofitems numeric(6);
link varchar;
onerow RECORD;
```

對整數範圍反覆運算的 FOR 迴圈的迴圈變數，自動宣告為整數變數。

DEFAULT 子句 (如有提供) 指定進入區塊時指派給變數的初始值。如果未提供 DEFAULT 子句，變數會初始化為 SQL NULL 值。CONSTANT 選項防止指派變數，因此在整個區塊內，值維持固定。如果

指定 NOT NULL，則指派空值會導致執行時間錯誤。所有宣告為 NOT NULL 的變數必須指定非空值的預設值。

每次進入區塊時會評估預設值。例如，假設指派 now() 給 timestamp 類型的變數，則變數會具有目前函數呼叫的時間，而非預先編譯函數時的時間。

```
quantity INTEGER DEFAULT 32;
url VARCHAR := 'http://mysite.com';
user_id CONSTANT INTEGER := 10;
```

refcursor 資料類型是預存程序內的游標變數的資料類型。預存程序可以傳回 refcursor 值。如需詳細資訊，請參閱 傳回結果集。

別名宣告

如果預存程序的簽章省略引數名稱，您可以宣告該引數的別名。

```
name ALIAS FOR $n;
```

內建變數

支援下列內建變數：

- FOUND
- SQLSTATE
- SQLERRM
- GET DIAGNOSTICS integer_var := ROW_COUNT;

FOUND 是布林值類型的特殊變數。在每個程序呼叫內，FOUND 剛開始為 false。FOUND 由下列類型的陳述式來設定：

- SELECT INTO

  將 FOUND 設為 true 表示傳回一列，設為 false 表示未傳回任何列。
- UPDATE、INSERT 和 DELETE

  將 FOUND 設為 true 表示至少一列受影響，設為 false 表示不影響任何列。

- FETCH

  將 FOUND 設為 true 表示傳回一列，設為 false 表示未傳回任何列。

- FOR 陳述式

  將 FOUND 設為 true 表示 FOR 陳述式反覆運算一或多次，否則為設為 false。這適用於 FOR 陳述式的所有三種變體：整數 FOR 迴圈、記錄集 FOR 迴圈，以及動態記錄集 FOR 迴圈。

  在 FOR 迴圈結束時設定 FOUND。在迴圈執行期內，FOR 陳述式不會修改 FOUND。不過，迴圈主體內的其他陳述式執行可能變更它。

下列顯示一個範例。

```
CREATE TABLE employee(empname varchar);
CREATE OR REPLACE PROCEDURE show_found()
AS  $$
DECLARE
  myrec record;
BEGIN
  SELECT INTO myrec * FROM employee WHERE empname = 'John';
  IF NOT FOUND THEN
    RAISE EXCEPTION 'employee John not found';
  END IF;
END;
$$ LANGUAGE plpgsql;
```

在例外處理常式內，特殊變數 SQLSTATE 包含錯誤碼，對應於已引發的例外狀況。特殊變數 SQLERRM 包含與例外狀況相關聯的錯誤訊息。這些變數在例外處理常式外未定義，如果使用，則會顯示錯誤。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE sqlstate_sqlerrm() AS
$$
BEGIN
  UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
  EXECUTE 'select invalid';
  EXCEPTION WHEN OTHERS THEN
  RAISE INFO 'error message SQLERRM %', SQLERRM;
  RAISE INFO 'error message SQLSTATE %', SQLSTATE;
END;
```

```
$$ LANGUAGE plpgsql;
```

ROW_COUNT 與 GET DIAGNOSTICS 命令搭配使用。它顯示傳送到 SQL 引擎的最後一個 SQL 命令所處理的列數。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE sp_row_count() AS
$$
DECLARE
  integer_var int;
BEGIN
  INSERT INTO tbl_row_count VALUES(1);
  GET DIAGNOSTICS integer_var := ROW_COUNT;
  RAISE INFO 'rows inserted = %', integer_var;
END;
$$ LANGUAGE plpgsql;
```

記錄類型

RECORD 類型不是真的資料類型，只是預留位置。記錄類型變數採用 SELECT 或 FOR 命令期間指派給它們的列的實際列結構。記錄變數在每次被指派一個值時可能變更子結構。記錄變數在第一次被指派之前沒有子結構。嘗試存取其中的欄位會擲出執行時間錯誤。

```
name RECORD;
```

下列顯示一個範例。

```
CREATE TABLE tbl_record(a int, b int);
INSERT INTO tbl_record VALUES(1, 2);
CREATE OR REPLACE PROCEDURE record_example()
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
BEGIN
  FOR rec IN SELECT a FROM tbl_record
  LOOP
    RAISE INFO 'a = %', rec.a;
  END LOOP;
END;
```

```
$$;
```

支援的 PL/pgSQL 陳述式

PL/pgSQL 陳述式以程序性建構 (包括迴圈和條件式表達式) 來擴增 SQL 命令，以控制邏輯流程。可使用大部分 SQL 命令，包括資料處理語言 (DML) (例如 COPY、UNLOAD 和 INSERT) 和資料定義語言 (DDL) (例如 CREATE TABLE)。如需完整 SQL 命令的清單，請參閱 SQL 命令。此外，Amazon Redshift 還支援下列 PL/pgSQL 陳述式。

主題

指派

指派陳述式可指派值給變數。表達式必須傳回單一值。

```
identifier := expression;
```

也接受使用非標準的 = 來指派 (而不是 :=)。

如果表達式的資料類型不符合變數的資料類型，或變數具有大小或精確度，則會隱含轉換結果值。

如下列範例所示。

```
customer_number := 20;
```

```
tip := subtotal * 0.15;
```

SELECT INTO

SELECT INTO 陳述式將多欄 (但只有一列) 的結果指派給一個記錄變數或一個純量變數清單。

```
SELECT INTO target select_expressions FROM ...;
```

在上述語法中，*target* 可以是記錄變數，或簡單變數和記錄欄位的逗號分隔清單。*select_expressions* 清單和命令的剩餘部分與一般 SQL 中相同。

如果以變數清單作為 *target*，選取的值必須完全符合目標的結構，否則會發生執行時間錯誤。當記錄變數為目標時，它本身會自動設定為查詢結果欄的列類型。

INTO 子句幾乎可出現在 SELECT 陳述式中的任意處。通常就出現在 SELECT 子句之後，或就在 FROM 子句之前。亦即，就出現在 *select_expressions* 清單之前或之後。

如果查詢未傳回任何列，NULL 值會指派給 *target*。如果查詢傳回多列，第一列會指派給 *target*，其餘捨棄。除非陳述式包含 ORDER BY，否則無法確定第一列。

若要判斷指派是否傳回至少一列，請使用特殊的 FOUND 變數。

```
SELECT INTO customer_rec * FROM cust WHERE custname = lname;
IF NOT FOUND THEN
  RAISE EXCEPTION 'employee % not found', lname;
END IF;
```

若要測試記錄結果是否為空值，您可以使用 IS NULL 條件。無法判斷是否已捨棄任何其他列。下列範例處理未傳回任何列的情況。

```
CREATE OR REPLACE PROCEDURE select_into_null(return_webpage OUT varchar(256))
AS $$
DECLARE
  customer_rec RECORD;
BEGIN
  SELECT INTO customer_rec * FROM users WHERE user_id=3;
  IF customer_rec.webpage IS NULL THEN
    -- user entered no webpage, return "http://"
    return_webpage = 'http://';
  END IF;
END;
```

```
$$ LANGUAGE plpgsql;
```

無操作

無操作陳述式 (NULL;) 是不執行任何動作的預留位置陳述式。無操作陳述式可以表示 IF-THEN-ELSE 鏈的一個分支是空的。

```
NULL;
```

動態 SQL

若要產生動態命令，以便每次從 PL/pgSQL 預存程序執行時，都可涉及不同的資料表或不同的資料類型，請使用 EXECUTE 陳述式。

```
EXECUTE command-string [ INTO target ];
```

在上述語法中，*command-string* 是產生字串 (文字類型) 的表達式，而此字串包含要執行的命令。此 *command-string* 值會傳送到 SQL 引擎。在命令字串上不會替換 PL/pgSQL 變數。您必須在建構命令字串時插入變數的值。

> **Note**
>
> 您無法從動態 SQL 內使用 COMMIT 和 ROLLBACK 陳述式。如需在預存程序內使用 COMMIT 和 ROLLBACK 陳述式的相關資訊，請參閱管理交易。

使用動態命令時，您通常需要處理單引號逸出。建議使用 $ 符號引用來圍住函數主體中放在引號內的固定字串。在建構的查詢中要插入的動態值需要特別處理，因為動態值本身可能包含引號。下列範例對整個函數採用 $ 符號引用，因此引號不需要加倍。

```
EXECUTE 'UPDATE tbl SET '
  || quote_ident(colname)
  || ' = '
  || quote_literal(newvalue)
  || ' WHERE key = '
  || quote_literal(keyvalue);
```

上述範例顯示函數 quote_ident(text) 和 quote_literal(text)。此範例將包含欄和資料表識別碼的變數傳給 quote_ident 函數。也將包含所建構命令中常值字串的變數傳給 quote_literal

函數。這兩個函數都採取適當步驟，傳回分別以雙引號或單引號括住的輸入文字，並適當地逸出任何內嵌的特殊字元。

$ 符號引用僅適用於括住固定文字。請勿將上述範例寫成下列格式。

```
EXECUTE 'UPDATE tbl SET '
  || quote_ident(colname)
  || ' = $$'
  || newvalue
  || '$$ WHERE key = '
  || quote_literal(keyvalue);
```

不能這樣做，因為如果 newvalue 的內容碰巧包含 $$，則範例會失敗。您可能選擇的任何其他 $ 符號引用分隔符號也有相同問題。若要安心地括住無法事先得知的文字，請使用 quote_literal 函數。

傳回

RETURN 陳述式從預存程序傳回到發起人。

```
RETURN;
```

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE return_example(a int)
AS $$
BEGIN
  FOR b in 1..10 LOOP
    IF b < a THEN
      RAISE INFO 'b = %', b;
    ELSE
      RETURN;
    END IF;
  END LOOP;
END;
$$ LANGUAGE plpgsql;
```

條件式：IF

在 Amazon Redshift 所使用的 PL/pgSQL 語言中，IF 條件式陳述式有下列形式：

- IF ... THEN

```
IF boolean-expression THEN
  statements
END IF;
```

下列顯示一個範例。

```
IF v_user_id <> 0 THEN
  UPDATE users SET email = v_email WHERE user_id = v_user_id;
END IF;
```

- IF ... THEN ... ELSE

```
IF boolean-expression THEN
  statements
ELSE
  statements
END IF;
```

下列顯示一個範例。

```
IF parentid IS NULL OR parentid = ''
THEN
  return_name = fullname;
  RETURN;
ELSE
  return_name = hp_true_filename(parentid) || '/' || fullname;
  RETURN;
END IF;
```

- IF ... THEN ... ELSIF ... THEN ... ELSE

關鍵字 ELSIF 也可以拼寫為 ELSEIF。

```
IF boolean-expression THEN
  statements
[ ELSIF boolean-expression THEN
  statements
[ ELSIF boolean-expression THEN
  statements
    ...] ]
[ ELSE
```

```
  statements ]
END IF;
```

下列顯示一個範例。

```
IF number = 0 THEN
  result := 'zero';
ELSIF number > 0 THEN
  result := 'positive';
ELSIF number < 0 THEN
  result := 'negative';
ELSE
  -- the only other possibility is that number is null
  result := 'NULL';
END IF;
```

條件式：CASE

在 Amazon Redshift 所使用的 PL/pgSQL 語言中，CASE 條件式陳述式有下列形式：

- 簡單 CASE

```
CASE search-expression
WHEN expression [, expression [ ... ]] THEN
  statements
[ WHEN expression [, expression [ ... ]] THEN
  statements
  ... ]
[ ELSE
  statements ]
END CASE;
```

簡單 CASE 陳述式可根據運算元等式而有條件地執行。

search-expression 值會評估一次，然後連續地與 WHEN 子句中的每個 *expression* 相比較。如果發現相符，則對應的 *statements* 會執行，接著控制權就移轉到 END CASE 之後的下一個陳述式。不會評估後續的 WHEN expressions。如果找不到相符，則 ELSE *statements* 會執行。不過，如果 ELSE 不存在，則會引發 CASE_NOT_FOUND 例外狀況。

下列顯示一個範例。

```
CASE x
WHEN 1, 2 THEN
  msg := 'one or two';
ELSE
  msg := 'other value than one or two';
END CASE;
```

- 搜尋的 CASE

```
CASE
WHEN boolean-expression THEN
  statements
[ WHEN boolean-expression THEN
  statements
  ... ]
[ ELSE
  statements ]
END CASE;
```

搜尋形式的 CASE 可根據布林值表達式的真實性而有條件地執行。

每個 WHEN 子句的 *boolean-expression* 會依次評估，直到發現產生 true 為止。然後對應的陳述式會執行，接著控制權就移轉到 END CASE 之後的下一個陳述式。不會評估後續的 WHEN *expressions*。如果找不到 true 結果，則 ELSE *statements* 會執行。不過，如果 ELSE 不存在，則會引發 CASE_NOT_FOUND 例外狀況。

下列顯示一個範例。

```
CASE
WHEN x BETWEEN 0 AND 10 THEN
  msg := 'value is between zero and ten';
WHEN x BETWEEN 11 AND 20 THEN
  msg := 'value is between eleven and twenty';
END CASE;
```

迴圈

在 Amazon Redshift 所使用的 PL/pgSQL 語言中，迴圈陳述式有下列形式：

- 簡單迴圈

```
[<<label>>]
LOOP
  statements
END LOOP [ label ];
```

簡單迴圈定義無條件的迴圈，將會無限期重複，直到由 EXIT 或 RETURN 陳述式終止為止。巢狀迴圈內的 EXIT 和 CONTINUE 陳述式可使用選用標籤，以指定 EXIT 和 CONTINUE 陳述式所指的迴圈。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE simple_loop()
LANGUAGE plpgsql
AS $$
BEGIN
  <<simple_while>>
  LOOP
    RAISE INFO 'I am raised once';
    EXIT simple_while;
    RAISE INFO 'I am not raised';
  END LOOP;
  RAISE INFO 'I am raised once as well';
END;
$$;
```

- 結束迴圈

```
EXIT [ label ] [ WHEN expression ];
```

如果 *label* 不存在，最內層迴圈會終止，END LOOP 之後的陳述式會接下去執行。如果 *label* 存在，它必須是巢狀迴圈或區塊的目前或某個外層的標籤。然後，具名迴圈或區塊會終止，控制權會延續到迴圈或區塊相對應 END 之後的陳述式。

如果指定 WHEN，只有在 *expression* 為 true 時，迴圈才會結束。否則，控制權會移轉到 EXIT 之後的陳述式。

您可以對所有類型的迴圈使用 EXIT；不限於用於無條件的迴圈。

與 BEGIN 區塊一起使用時，EXIT 會將控制權移轉到區塊結束之後的下一個陳述式。為此，必須使用標籤。無標籤的 EXIT 絕不可能有對稱的 BEGIN 區塊。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE simple_loop_when(x int)
LANGUAGE plpgsql
AS $$
DECLARE i INTEGER := 0;
BEGIN
  <<simple_loop_when>>
  LOOP
    RAISE INFO 'i %', i;
    i := i + 1;
    EXIT simple_loop_when WHEN (i >= x);
  END LOOP;
END;
$$;
```

- 繼續迴圈

```
CONTINUE [ label ] [ WHEN expression ];
```

如果未提供 *label*，執行會跳到最內層迴圈的下一次反覆運算。亦即會略過迴圈主體中剩餘的所有陳述式。然後，控制權會返回到迴圈控制表達式 (如果有的話)，以決定是否需要再一次迴圈反覆運算。如果 *label* 存在，它指定繼續執行的迴圈的標籤。

如果指定 WHEN，只有在 *expression* 為 true 時，迴圈的下一次反覆運算才會開始。否則，控制權會移轉到 CONTINUE 之後的陳述式。

您可以對所有類型的迴圈使用 CONTINUE；不限於用於無條件的迴圈。

```
CONTINUE mylabel;
```

- WHILE 迴圈

```
[<<label>>]
WHILE expression LOOP
  statements
END LOOP [ label ];
```

只要 *boolean-expression* 評估為 true，WHILE 陳述式會重複一連串陳述式。就在進入迴圈主體之前會檢查表達式。

下列顯示一個範例。

```
WHILE amount_owed > 0 AND gift_certificate_balance > 0 LOOP
  -- some computations here
END LOOP;

WHILE NOT done LOOP
  -- some computations here
END LOOP;
```

- FOR 迴圈 (整數變體)

```
[<<label>>]
FOR name IN [ REVERSE ] expression .. expression LOOP
  statements
END LOOP [ label ];
```

FOR 迴圈 (整數變數) 建立迴圈對整數值範圍反覆運算。變數名稱會自動定義為整數類型，並且只在迴圈內結束。迴圈內會忽略變數名稱的任何現有定義。定義範圍下限和上限的兩個表達式會在進入迴圈時評估一次。如果您指定 REVERSE，則每一次反覆運算後會減去間距值，而不是相加。

如果下限大於上限 (或在 REVERSE 情況下是小於)，迴圈主體不會執行。不會引發錯誤。

如果標籤附加到 FOR 迴圈，則您可以使用該標籤，以限定名稱來參考整數迴圈變數。

下列顯示一個範例。

```
FOR i IN 1..10 LOOP
  -- i will take on the values 1,2,3,4,5,6,7,8,9,10 within the loop
END LOOP;

FOR i IN REVERSE 10..1 LOOP
  -- i will take on the values 10,9,8,7,6,5,4,3,2,1 within the loop
END LOOP;
```

- FOR 迴圈 (結果集變體)

```
[<<label>>]
FOR target IN query LOOP
  statements
END LOOP [ label ];
```

target 是記錄變數，或純量變數的逗號分隔清單。從查詢產生的每一列會連續指派給目標，而每一列會執行一次迴圈主體。

FOR 迴圈 (結果集變體) 可讓預存程序逐一查看查詢的結果，並相應地操作該資料。

下列顯示一個範例。

```
CREATE PROCEDURE cs_refresh_reports() AS $$
DECLARE
  reports RECORD;
BEGIN
  FOR reports IN SELECT * FROM cs_reports ORDER BY sort_key LOOP
    -- Now "reports" has one record from cs_reports
    EXECUTE 'INSERT INTO ' || quote_ident(reports.report_name) || ' ' ||
 reports.report_query;
  END LOOP;
  RETURN;
END;
$$ LANGUAGE plpgsql;
```

- FOR 迴圈搭配動態 SQL

```
[<<label>>]
FOR record_or_row IN EXECUTE text_expression LOOP
  statements
END LOOP;
```

FOR 迴圈搭配動態 SQL 可讓預存程序逐一查看動態查詢的結果，並相應地操作該資料。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE for_loop_dynamic_sql(x int)
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  query text;
BEGIN
  query := 'SELECT * FROM tbl_dynamic_sql LIMIT ' || x;
  FOR rec IN EXECUTE query
  LOOP
```

```
    RAISE INFO 'a %', rec.a;
  END LOOP;
END;
$$;
```

游標

您可以設定游標，而不要一次執行整個查詢。cursor 封裝查詢，每次在查詢結果中讀取幾列。這麼做的一個理由是當結果包含大量的列時，避免記憶體溢位。另一個理由是為了傳回預存程序已建立之游標的參考，此游標可讓發起人讀取列。此方法可以很有效率地從預存程序傳回很大的列集。

若要在 NONATOMIC 預存程序中使用游標，請將游標迴圈置於 START TRANSACTION...COMMIT 之間。

若要設定游標，首先要宣告游標變數。在 PL/pgSQL 中存取游標完全是透過游標變數，該變數一定是特殊資料類型 `refcursor`。`refcursor` 資料類型只是保留游標的參考。

您可以將變數宣告為 `refcursor` 類型，以建立游標變數。或者，您可以使用如下的游標宣告語法。

```
name CURSOR [ ( arguments ) ] FOR query ;
```

在上述語法中，*arguments* (如有指定) 是 *name datatype* 組的逗號分隔清單，每一組定義要由 *query* 中的參數值所取代的名稱。稍後開啟游標時指定用來替換這些名稱的實際值。

如下列範例所示。

```
DECLARE
  curs1 refcursor;
  curs2 CURSOR FOR SELECT * FROM tenk1;
  curs3 CURSOR (key integer) IS SELECT * FROM tenk1 WHERE unique1 = key;
```

這三個變數全部都是資料類型 `refcursor`，但第一個可用於任何查詢。相反地，第二個已繫結完整指定的查詢，而最後一個已繫結參數化查詢。開啟游標時，`key` 值會換成整數參數值。變數 `curs1` 可說成未繫結，因為沒有繫結至任何特定查詢。

游標必須先開始，才能用來擷取列。PL/pgSQL 有三種形式的 OPEN 陳述式，其中兩種使用未繫結的游標變數，第三種使用繫結的游標變數：

- 開啟以選取：開啟游標變數，並給予要執行的指定查詢。游標不能已開啟。另外，還必須已宣告為未繫結的游標 (亦即，宣告為簡單的 refcursor 變數)。SELECT 查詢的處理方式與 PL/pgSQL 中的其他 SELECT 陳述式相同。

```
OPEN cursor_name FOR SELECT ...;
```

下列顯示一個範例。

```
OPEN curs1 FOR SELECT * FROM foo WHERE key = mykey;
```

- 開啟以執行：開啟游標變數，並給予要執行的指定查詢。游標不能已開啟。另外，還必須已宣告為未繫結的游標 (亦即，宣告為簡單的 refcursor 變數)。將查詢指定為字串表達式的方式與 EXECUTE 命令中相同。此方法很有彈性，可讓查詢隨著每一次執行而變化。

```
OPEN cursor_name FOR EXECUTE query_string;
```

下列顯示一個範例。

```
OPEN curs1 FOR EXECUTE 'SELECT * FROM ' || quote_ident($1);
```

- 開啟繫結的游標：這種 OPEN 用於開啟已在宣告時繫結查詢的游標變數。游標不能已開啟。僅當游標宣告為接受引數時，實際引數值表達式的清單才必須出現。查詢中會替換這些值。

```
OPEN bound_cursor_name [ ( argument_values ) ];
```

下列顯示一個範例。

```
OPEN curs2;
OPEN curs3(42);
```

開啟游標後，您可以利用下述的陳述式來使用它。這些陳述式不一定要出現在開啟游標的同一預存程序中。您可以從預存程序傳回 refcursor 值，並讓發起人繼續操作游標。交易結束時，所有入口會隱含關閉。因此，只有在交易結束後，您才可以使用 refcursor 值來參考開啟的游標。

- FETCH 從游標中將下一列擷取到目標。此目標可以是列變數、記錄變數，或簡單變數的逗號分隔清單，如同 SELECT INTO 一樣。如同 SELECT INTO 一樣，您可以檢查特殊變數 FOUND，查明是否已取得一列。

```
FETCH cursor INTO target;
```

下列顯示一個範例。

```
FETCH curs1 INTO rowvar;
```

- CLOSE 會將已開啟的游標的基礎入口關閉。在交易結束之前，您可以使用此陳述式來提早釋放資源。您也可以使用此陳述式來釋放游標變數，供再次開啟。

```
CLOSE cursor;
```

下列顯示一個範例。

```
CLOSE curs1;
```

RAISE

使用 RAISE level 陳述式來報告訊息和引發錯誤。

```
RAISE level 'format' [, variable [, ...]];
```

可能的等級包括 NOTICE、INFO、LOG、WARNING 和 EXCEPTION。EXCEPTION 會引發錯誤，通常會取消目前的交易。其他等級只產生不同優先等級的訊息。

在格式字串內，% 會換成下一個選用引數的字串表示法。撰寫 %% 以發出常值 %。目前，選用引數必須是簡單變數，不是表達式，而格式必須是簡單字串常值。

在下列範例中，v_job_id 的值會取代字串中的 %。

```
RAISE NOTICE 'Calling cs_create_job(%)', v_job_id;
```

使用 RAISE 陳述式重新擲出例外狀況處理區塊捕獲的例外狀況。這個陳述式僅在 NONATOMIC 模式預存程序的例外狀況處理區塊中有效。

```
RAISE;
```

交易控制

您可以在 Amazon Redshift 所使用的 PL/pgSQL 語言中使用交易控制陳述式。如需在預存程序內使用 COMMIT、ROLLBACK 和 TRUNCATE 陳述式的相關資訊，請參閱管理交易。

在 NONATOMIC 模式預存程序中，使用 START TRANSACTION 啟動交易區塊。

```
START TRANSACTION;
```

> **Note**
>
> PL/pgSQL 陳述式 START TRANSACTION 與 SQL 命令 START TRANSACTION 有以下不同：
>
> - 在預存程序中，START TRANSACTION 與 BEGIN 不同義。
> - PL/pgSQL 陳述式不支援選擇性的隔離層級和存取許可關鍵字。

在 Amazon Redshift 中建立具體化視觀表

在資料倉儲環境中，應用程式通常必須在大型資料表上執行複雜的查詢。例如，SELECT 陳述式會在包含數十億個資料列的資料表上執行多資料表的聯結和彙總。處理這些查詢相當耗費系統資源，且運算結果也會耗費時間。

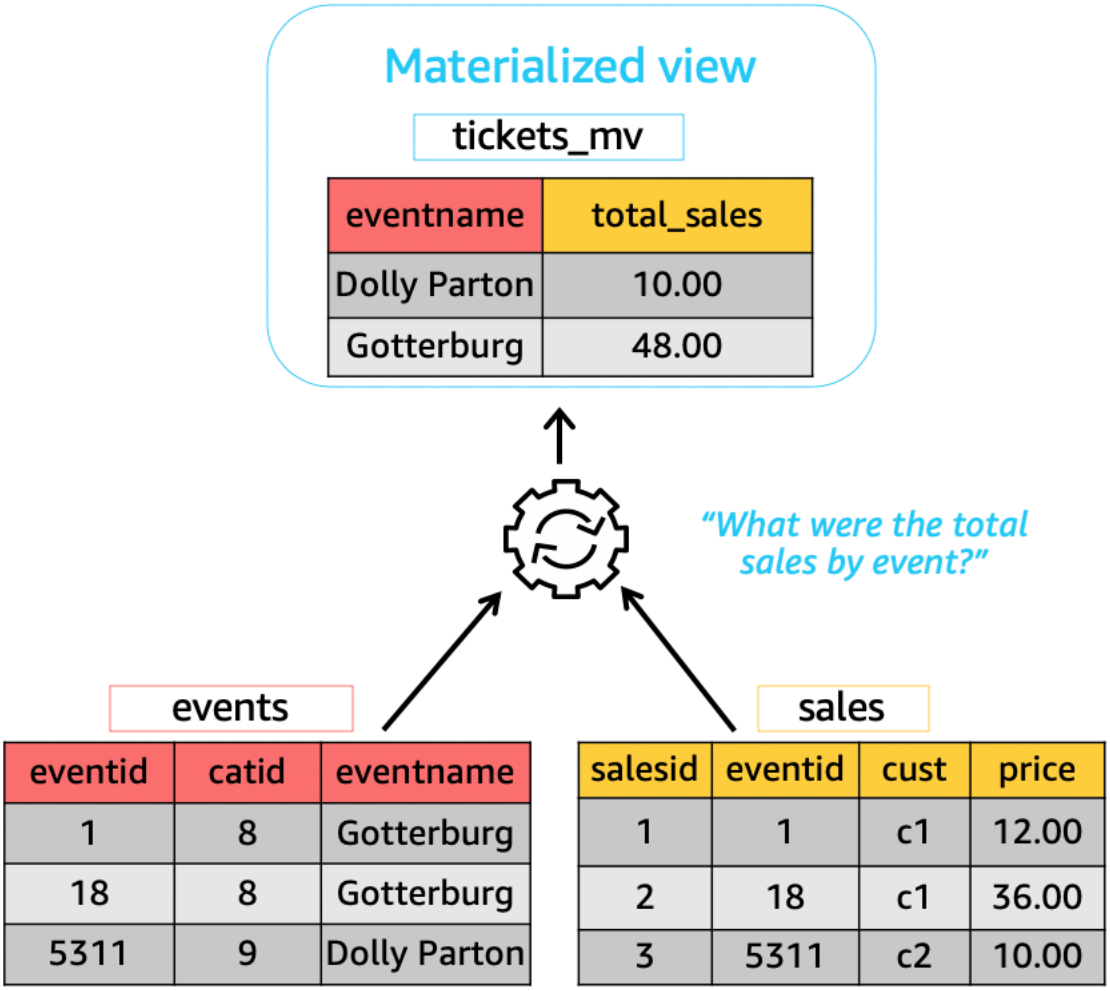
Amazon Redshift 中的具體化視觀表提供處理這些問題的方式。具體化視觀表包含預先運算的結果集，該結果集是以對一或多個基底資料表進行的 SQL 查詢為基礎。您可以使用與查詢資料庫中其他資料表或檢視相同的方式，發出 SELECT 陳述式來查詢具體化視觀表。Amazon Redshift 會從具體化視觀表傳回預先計算的結果，完全不需要存取基底資料表。從使用者的觀點而言，傳回查詢結果的速度會比從基底資料表擷取相同的資料時要快上許多。

具體化視觀表在加速可預測和重複的查詢時特別有用。相較於對大型資料表 (例如彙整或多個聯結) 執行耗費資源的查詢，應用程式可以改為查詢具體化視觀表，並擷取預先運算的結果集。例如，假設使用一組查詢來填入儀表板 (例如 Amazon) 的案例 QuickSight。這種使用案例便相當適合使用具體化視觀表，因為查詢是可預期的，且會一再地重複進行。

您可以根據其他具體化視觀表來定義具體化視觀表。使用具體化視觀表上的具體化視觀表來擴充具體化視觀表的功能。在這種方法中，現有具體化視觀表的角色會與查詢用來擷取資料的基底資料表相同。

針對不同彙總或 GROUP BY 選項重複使用預先計算的聯結時，此方法特別有用。例如，結合客戶資訊 (包含數百萬個資料列) 與項目訂單詳細資訊 (包含數十億個資料列) 的具體化視觀表。這是重複按需計算的高成本查詢。您可以針對在此具體化視觀表之上建立的具體化視觀表使用不同的 GROUP BY 選項，並與其他資料表結合。這樣做可以節省計算時間，否則每次都要執行高成本的基礎聯結。STV_MV_DEPS 資料表會顯示具體化視觀表在其他具體化視觀表上的相依性。

當您建立具體化視觀表時，Amazon Redshift 會執行使用者指定的 SQL 陳述式，從基底資料表或資料表收集資料，並儲存結果集。下圖為 SQL 查詢使用兩個基礎資料表 `events` 和 `sales` 定義的具體化視觀表 `tickets_mv` 的簡介。



然後，您可以在查詢中使用這些具體化視觀表來加速查詢。此外，即使查詢未明確參照具體化視觀表，Amazon Redshift 也可以自動重寫這些查詢以使用具體化視觀表。當您無法將查詢變更為使用具體化視觀表時，自動重寫查詢在提升效能方面尤其強大。

若要更新具體化視觀表中的資料，您可以隨時使用 REFRESH MATERIALIZED VIEW 陳述式來手動重新整理具體化視觀表。Amazon Redshift 會識別在基底資料表或資料表中發生的變更，然後將這些變更套用到具體化視觀表。由於自動重寫查詢需要具體化視觀表維持在最新狀態，所以身為具體化視觀表擁有者，請務必在基礎資料表變更時重新整理具體化視觀表。

Amazon Redshift 提供了幾種讓具體化視觀圖保持在最新狀態的方法，以便自動重寫。您可以使用自動重新整理選項設定具體化視觀表，以在更新具體化視觀表的基礎資料表時重新整理具體化視觀表。此自動重新整理操作會在叢集資源可用時執行，以盡量減少對其他工作負載的干擾。由於自動重新整理排程取決於工作負載，因此您可以更好地控制 Amazon Redshift 重新整理具體化視觀表的時機。您可以使用 Amazon Redshift 排程器 API 和主控台整合來排程具體化視觀表的重新整理工作。如需查詢排程的相關資訊，請參閱在 Amazon Redshift 主控台上排程查詢。

當具體化視觀表中的 up-to-date 資料有服務等級協定 (SLA) 需求時，這樣做特別有用。您也可以手動重新整理任何可以自動重新整理的具體化視觀表。如需如何建立具體化視觀表的相關資訊，請參閱CREATE MATERIALIZED VIEW。

您可以發出 SELECT 陳述式來查詢具體化視觀表。如需如何查詢具體化視觀表的相關資訊，請參閱查詢具體化視觀表。結果集最終會隨著在基礎資料表中插入、更新和刪除資料而過時。您可以隨時重新整理具體化視觀表，以使用基礎資料表中的最新變更來更新它。如需如何重新整理具體化視觀表的相關資訊，請參閱REFRESH MATERIALIZED VIEW。

如需用於建立和管理具體化視觀表的 SQL 命令詳細資訊，請參閱下列命令主題：

- CREATE MATERIALIZED VIEW
- ALTER MATERIALIZED VIEW
- REFRESH MATERIALIZED VIEW
- DROP MATERIALIZED VIEW

如需用於監控具體化視觀表的系統資料表和檢視相關資訊，請參閱下列主題：

- STV_MV_INFO
- STL_MV_STATE
- SVL_MV_REFRESH_STATUS
- STV_MV_DEPS

主題

- 查詢具體化視觀表
- 自動查詢重寫以使用具體化視檢視
- 重新整理具體化視觀表
- 自動具體化視觀表
- 在具體化視觀表中使用使用者定義函數 (UDF)
- 串流擷取

查詢具體化視觀表

您可以透過參考作為資料來源的具體化視觀表名稱在任何 SQL 查詢中使用具體化視觀表，就如同資料表或標準檢視。

當查詢存取具體化視觀表時，其只會看見最近一次重新整理後儲存在具體化視觀表中的資料。因此，查詢可能無法看見具體化視觀表所對應基底資料表中所有最新的變更。

如果其他使用者希望查詢具體化視觀表，具體化視觀表的擁有者會將 SELECT 許可授予那些使用者。其他使用者不需要對基礎的基底資料表具備 SELECT 許可。具體化視觀表的擁有者也可以撤銷其他使用者的 SELECT 許可，防止他們查詢具體化視觀表。

如果具體化視觀表的擁有者不再擁有對基礎資料表的 SELECT 許可：

- 擁有者無法再查詢具體化視觀表。
- 其他對具體化視觀表擁有 SELECT 許可的使用者無法再查詢具體化視觀表。

以下範例會查詢 `tickets_mv` 具體化視觀表。如需用來建立具體化視觀表的 SQL 命令相關資訊，請參閱 CREATE MATERIALIZED VIEW。

```
SELECT sold
FROM tickets_mv
WHERE catgroup = 'Concerts';
```

因為查詢結果是預先運算的，因此無須存取基礎資料表 (`category`、`event` 和 `sales`)。Amazon Redshift 可以直接從 `tickets_mv` 傳回結果。

自動查詢重寫以使用具體化視檢視

您可以使用 Amazon Redshift 中具體化視觀表的自動查詢重寫功能，讓 Amazon Redshift 重寫查詢以使用具體化視觀表。這樣做可以加速查詢工作負載，即使是未明確參考具體化視觀表的查詢也一樣。當 Amazon Redshift 重寫查詢時，其只會使用最新的具體化視觀表。

使用須知

若要檢查查詢的自動重寫功能是否用於查詢，您可以檢查查詢計劃或 STL_EXPLAIN。下面顯示了 SELECT 陳述式和原始查詢計劃的 EXPLAIN 輸出。

```
SELECT catgroup, SUM(qtysold) AS sold
FROM category c, event e, sales s
WHERE c.catid = e.catid AND e.eventid = s.eventid
GROUP BY 1;

EXPLAIN
 XN HashAggregate  (cost=920021.24..920021.24 rows=1 width=35)
```

```
   ->  XN Hash Join DS_BCAST_INNER  (cost=440004.53..920021.22 rows=4 width=35)
         Hash Cond: ("outer".eventid = "inner".eventid)
         ->  XN Seq Scan on sales s  (cost=0.00..7.40 rows=740 width=6)
         ->  XN Hash  (cost=440004.52..440004.52 rows=1 width=37)
               ->  XN Hash Join DS_BCAST_INNER  (cost=0.01..440004.52 rows=1 width=37)
                     Hash Cond: ("outer".catid = "inner".catid)
                     ->  XN Seq Scan on event e  (cost=0.00..2.00 rows=200 width=6)
                     ->  XN Hash  (cost=0.01..0.01 rows=1 width=35)
                           ->  XN Seq Scan on category c  (cost=0.00..0.01 rows=1
 width=35)
```

下面顯示成功自動重寫後的 EXPLAIN 輸出。此輸出包括查詢計劃中具體化視觀表的掃描，其取代了原始查詢計劃的一部分。

```
* EXPLAIN
     XN HashAggregate  (cost=11.85..12.35 rows=200 width=41)
       ->  XN Seq Scan on mv_tbl__tickets_mv__0 derived_table1  (cost=0.00..7.90
 rows=790 width=41)
```

只有 up-to-date (新的) 具體化視觀表才會被視為自動重新寫入查詢，不論重新整理策略為何，例如 auto 動、排程或手動。因此，原始查詢返回 up-to-date 結果。在查詢中明確參照具體化視觀表時，Amazon Redshift 會存取具體化視觀表中目前儲存的資料。此資料可能不會反映具體化視觀表之基礎資料表的最新變更。

您可以使用在叢集 1.0.20949 版或更新版本上建立的具體化視觀表自動查詢重寫功能。

您可以藉由使用 SET mv_enable_aqmv_for_session to FALSE，停止工作階段層級上的自動查詢重寫功能。

限制

以下是針對具體化視觀表使用自動查詢重寫的限制。

- 自動查詢重寫適用於不參照或包含下列任何項目的具體化視觀表：
  - 子查詢
  - 左側、右側或完整外部聯結
  - 設定操作
  - 任何彙總函數，但 SUM、COUNT、MIN、MAX 與 AVG 除外。(這些是唯一可與自動查詢重寫搭配使用的彙總函數。)
  - 任何搭配 DISTINCT 的彙總函數

  - 任何範圍函數
  - SELECT DISTINCT 或 HAVING 子句
  - 外部資料表
  - 其他具體化視觀表
- 自動查詢重寫功能會重寫參照使用者定義 Amazon Redshift 資料表的 SELECT 查詢。Amazon Redshift 不會重寫以下查詢：
  - CREATE TABLE AS 陳述式
  - SELECT INTO 陳述式
  - 目錄或系統資料表的查詢
  - 具有外部聯結或 SELECT DISTINCT 的查詢
- 如果查詢未自動重寫，請檢查您是否擁有指定具體化視觀表的 SELECT 許可，且 mv_enable_aqmv_for_session 選項設定為 TRUE。

  您也可以藉由檢查 STV_MV_INFO，來檢查具體化視觀表是否符合自動重寫查詢的資格。如需詳細資訊，請參閱 STV_MV_INFO。

重新整理具體化視觀表

當您建立具體化視觀表時，其內容會反映當時基礎資料庫資料表或資料表的狀態。即使應用程式變更基底資料表中的資料，具體化視觀表中的資料仍會維持不變。若更新具體化視觀表中的資料，您可以隨時使用 `REFRESH MATERIALIZED VIEW` 陳述式來手動重新整理具體化視觀表。當您執行此陳述式時，Amazon Redshift 會識別在基底資料表或資料表中發生的變更，並將這些變更套用到具體化視觀表。

Amazon Redshift 有兩種策略可重新整理具體化視觀表：

- 在許多案例中，Amazon Redshift 可以執行累加式重新整理。在累加式重新整理中，Amazon Redshift 會快速識別自上次重新整理之後，基礎資料表中資料的變更，並更新具體化視觀表中的資料。定義具體化視觀表時，在查詢中使用的下列 SQL 建構模組上，可支援累加式重新整理。
  - 包含 SELECT、FROM、[INNER] JOIN、WHERE、GROUP BY、HAVING 子句的建構模組。
  - 包含彙總的建構模組，如 SUM、MIN、MAX、AVG 和 COUNT。
  - 大部分的內建 SQL 函數 (特別是不可變的 SQL 函數) 有鑑於此，都會給與相同的輸入引數並始終產生相同的輸出。

  以資料共用資料表為基礎的具體化視觀表也支援累加式重新整理。

- 如果無法進行累加式更新，則 Amazon Redshift 會執行完全的重新整理。「完全重新整理」會重新執行基礎 SQL 陳述式，替換具體化視觀表中的所有資料。
- Amazon Redshift 會根據用來定義具體化視觀表的 SELECT 查詢，自動選擇具體化視觀表的重新整理方法。

重新整理具體化視觀表上的具體化視觀表並不是串聯處理。換句話說，假設您有一個依賴具體化視觀表 B 的具體化視觀表 A。在此情況下，當呼叫「重新整理具體化視觀表 A」時，會使用目前版本的 B 重新整理 A，即使 B 為也是如此 out-of-date。要使 A 完全保持最新狀態，請在重新整理 A 之前，先在單獨的交易中重新整理 B。

以下範例顯示如何以程式設計方式，為具體化視觀表建立完整重新整理計劃。若要重新整理具體化視觀表 v，請先重新整理具體化視觀表 u 若要重新整理具體化視觀表 v，請先重新整理具體化視觀表 u，然後再重新整理具體化視觀表 v。

```
CREATE TABLE t(a INT);
CREATE MATERIALIZED VIEW u AS SELECT * FROM t;
CREATE MATERIALIZED VIEW v AS SELECT * FROM u;
CREATE MATERIALIZED VIEW w AS SELECT * FROM v;

WITH RECURSIVE recursive_deps (mv_tgt, lvl, mv_dep) AS
( SELECT trim(name) as mv_tgt, 0 as lvl, trim(ref_name) as mv_dep
  FROM stv_mv_deps
  UNION ALL
  SELECT R.mv_tgt, R.lvl+1 as lvl, trim(S.ref_name) as mv_dep
  FROM stv_mv_deps S, recursive_deps R
  WHERE R.mv_dep = S.name
)

SELECT mv_tgt, mv_dep from recursive_deps
ORDER BY mv_tgt, lvl DESC;

 mv_tgt | mv_dep
--------+--------
 v      | u
 w      | u
 w      | v
(3 rows)
```

下列範例顯示當您在相依具體化視觀表的具體化視觀表上執行「重新整理具體化視 out-of-date 觀表」時的資訊訊息。

```
create table a(a int);
```

```
create materialized view b as select * from a;
```

```
create materialized view c as select * from b;
```

```
insert into a values (1);
```

```
refresh materialized view c;

INFO:  Materialized view c is already up to date.  However, it depends on another
 materialized view that is not up to date.
```

```
REFRESH MATERIALIZED VIEW b;
INFO:  Materialized view b was incrementally updated successfully.
```

```
REFRESH MATERIALIZED VIEW c;
INFO:  Materialized view c was incrementally updated successfully.
```

Amazon Redshift 目前對於具體化視觀表的累加式重新整理有下列限制。

對於使用下列 SQL 元素搭配查詢定義的具體化視觀表，Amazon Redshift 不支援累加式重新整理：

- OUTER JOIN (RIGHT、LEFT 或 FULL)。
- 集合操作 UNION、INTERSECT、EXCEPT 和 MINUS。
- 彙整函數
  MEDIAN、PERCENTILE_CONT、LISTAGG、STDDEV_SAMP、STDDEV_POP、APPROXIMATE COUNT、APPROXIMATE PERCENTILE 及位元彙整函數。

  > **Note**
  >
  > 支援 COUNT、SUM 和 AVG 彙總函數。

- DISTINCT 彙整函數，例如 DISTINCT COUNT、DISTINCT SUM 等。
- 視窗函數。
- 使用暫存資料表進行查詢最佳化的查詢，例如最佳化通用子運算式。

- 子查詢。
- 在定義具體化視觀表的查詢中，參照下列查詢中的格式的外部資料表。
  - Delta Lake
  - Hudi

  對於使用上述格式以外的格式定義的具體化視觀表，預覽軌道上支援累加式重新整理。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的建立預覽叢集。如需設定預覽工作群組的詳細資訊，請參閱 Amazon Redshift 管理指南中的建立預覽工作群組。

自動重新整理具體化視觀表

使用建立或變更具體化視圖以使用自動重新整理選項時，Amazon Redshift 可以使用其基礎表格中的 up-to-date 資料自動重新整理具體化視觀表。Amazon Redshift 會在基本資料表變更後盡快自動重新整理具體化視觀表。

為了在盡量不影響叢集內作用中工作負載的情況下完成重新整理最重要的具體化視觀表，Amazon Redshift 會考慮多個因素。這些因素包括目前的系統負載、重新整理所需的資源、可用的叢集資源，以及使用具體化視觀表的頻率。

Amazon Redshift 會優先考慮您的工作負載而非自動重新整理，並且可以停止自動重新整理以保留使用者工作負載的效能。此方法可能會延遲重新整理部分具體化視觀表。在某些情況下，您的具體化視觀表可能需要更具決定性的重新整理行為。如果是這樣，請考慮使用手動重新整理，如 REFRESH MATERIALIZED VIEW 所述，或透過 Amazon Redshift 排程器 API 操作或主控台來使用排程重新整理。

您可以使用 CREATE MATERIALIZED VIEW 來設定具體化視觀表的自動重新整理。您也可以使用 AUTO REFRESH 子句自動重新整理具體化視觀表。如需建立具體化視觀表的相關資訊，請參閱 CREATE MATERIALIZED VIEW。您可以使用 ALTER MATERIALIZED VIEW 來開啟目前具體化視觀表的自動重新整理。

重新整理具體化視觀表時應考慮以下項目：

- 即使您尚未啟用具體化視觀表的自動重新整理，您仍然可以使用 REFRESH MATERIALIZED VIEW 命令明確地重新整理具體化視觀表。
- Amazon Redshift 不會自動重新整理外部資料表上定義的具體化視觀表。
- 對於重新整理狀態，您可以檢查 SVL_MV_REFRESH_STATUS，該狀態會記錄使用者啟動或自動重新整理的查詢。

- 若要在僅限重新計算的具體化視觀表上執行 REFRESH，請確定您擁有結構描述的 CREATE 許可。如需詳細資訊，請參閱 GRANT。

自動具體化視觀表

具體化視觀表是改善 Amazon Redshift 中查詢效能的強大工具。其透過儲存預先計算的結果集來做到這一點。類似的查詢不必每次都重新執行相同的邏輯，因為它們可以從現有的結果集擷取記錄。開發人員和分析師會在分析工作負載之後建立具體化視觀表，以判斷哪些查詢可以受益，以及每個具體化視觀表的維護成本是否值得。隨著工作負載的增長或變化，這些具體化視觀表必須經過審查，以確保它們繼續提供實際的效能優勢。

Redshift 中的自動具體化視觀表 (AutoMV) 功能提供與使用者建立的具體化視觀表相同的效能優勢。Amazon Redshift 會使用機器學習持續監控工作負載，並在有益的情況下建立新的具體化視觀表。AutoMV 會平衡建立和維持具體化視觀表最新狀態的成本，以及查詢延遲的預期效益。系統也會監控先前建立的 AutoMV，並在它們不再有利時將其捨棄。

AutoMV 行為和功能與使用者建立的具體化視觀表相同。其使用相同的準則和限制以自動且累加的方式進行重新整理。就像使用者建立的具體化視觀表一樣，自動查詢重寫以使用具體化視檢視 會識別可受益於系統建立之 AutoMV 的查詢。它會自動重寫這些查詢以使用 AutoMV，從而改善查詢效能。開發人員不需要修改查詢即可利用 AutoMV。

> **Note**
>
> 自動具體化視觀表會間歇性重新整理。重寫為使用 AutoMV 的查詢永遠會傳回最新的結果。當 Redshift 偵測到資料不是最新的時候，就不會將查詢重寫為從自動具體化視觀表讀取。相反地，查詢會從基底資料表中選取最新資料。

任何具有重複使用之查詢的工作負載都可以受益於 AutoMV。常用案例包括：

- 儀表板 - 儀表板廣泛用於提供關鍵業務指標 (KPI)、事件、趨勢和其他指標的快速檢視。它們通常具有包含圖表和資料表的通用版面配置，但會顯示不同的檢視來用於篩選或維度選取操作 (例如向下鑽研)。儀表板通常有一組共同的查詢，以不同的參數重複使用。儀表板查詢可以從自動具體化視觀表中獲益匪淺。
- 報告 - 報告查詢可以安排在不同的頻率下進行，根據業務需求和報告的類型。此外，它們可以是自動或隨需形式。報告查詢的一個共同特徵是可以長時間執行且屬於資源密集型。使用 AutoMV 時，這些查詢不需要在每次執行時重新計算，這會減少 Redshift 中每個查詢的執行期和資源使用率。

若要關閉自動具體化視觀表，請將 auto_mv 參數群組更新為 false。如需詳細資訊，請參閱《Amazon Redshift 叢集管理指南》中的 Amazon Redshift 參數群組。

自動具體化視觀表的 SQL 範圍和考量

- 自動具體化視觀表可以由查詢或子查詢啟動和建立，前提是它包含 GROUP BY 子句或下列其中一個彙總函數：SUM、COUNT、MIN、MAX 或 AVG。但不能包含以下任何項目：
  - 左側、右側或完整外部聯結
  - SUM、COUNT、MIN、MAX 與 AVG 以外的彙總函數。(這些特定函式可與自動查詢重寫搭配使用。)
  - 任何包含 DISTINCT 的彙總函數
  - 任何範圍函數
  - SELECT DISTINCT 或 HAVING 子句
  - 其他具體化視觀表

  不保證符合準則的查詢會啟動自動具體化視觀表的建立。系統會根據其對工作負載的預期效益和要維護的資源成本 (包括系統要重新整理的成本)，決定要從哪些候選項目建立檢視。每個產生的具體化視觀表都可透過自動查詢重寫來使用。
- 即使 AutoMV 可能是由子查詢或集合運算子的個別支段啟動，產生的具體化視觀表也不會包含子查詢或集合運算子。
- 若要判斷 AutoMV 是否用於查詢，請檢視 EXPLAIN 計劃並在輸出中尋找 %_auto_mv_%。如需詳細資訊，請參閱 EXPLAIN。
- 外部資料表 (例如資料共用和聯合資料表) 不支援自動具體化視觀表。

自動具體化視觀表限制

下列是使用自動具體化視觀表的限制：

- AutoMV 的最大數目 - 叢集中每個資料庫的自動具體化視觀表限制為 200 個。
- 儲存空間和容量 - AutoMV 的一個重要特點是，它是使用備用背景週期來執行，以協助實現使用者工作負載不受影響的目的。如果叢集忙碌或儲存空間不足，AutoMV 會停止其活動。具體而言，在叢集總容量的 80% 時，不會建立新的自動具體化視觀表。在總容量的 90% 時，它們可能會被捨棄，以在不降低校能的情況下繼續使用者工作負載。如需決定叢集容量的相關資訊，請參閱 STV_NODE_STORAGE_CAPACITY。

自動具體化視觀表的計費

Amazon Redshift 的自動最佳化功能可建立和重新整理自動具體化視觀表。此程序的運算資源無須付費。自動具體化視觀表的儲存費用是以一般儲存費率計費。如需詳細資訊，請參閱 Amazon Redshift 定價。

其他資源

下列部落格文章提供有關自動具體化視觀表的進一步說明 其中詳細介紹了其建立、維護和捨棄方式。還說明了推動這些決策的基礎演算法：使用自動具體化視觀表最佳化 Amazon Redshift 查詢效能。

此影片從具體化視觀表的說明開始，並示範其如何改善效能及節省資源。然後，透過程序流程動畫和現場示範，提供自動具體化視觀表的深入說明。

在具體化視觀表中使用使用者定義函數 (UDF)

您可以在 Amazon Redshift 具體化視觀表中使用純量 UDF。在 python 或 SQL 中定義這些內容，並在具體化視觀表定義中加以參照。

在具體化視觀表中參照 UDF

下列程序顯示如何在具體化視觀表定義中使用執行簡單算術比較的 UDF。

1. 建立要在具體化視觀表定義中使用的資料表。

```
CREATE TABLE base_table (a int, b int);
```

2. 在 python 中建立一個純量使用者定義函數，該函數會傳回布林值，指出整數是否大於比較整數。

```
CREATE OR REPLACE FUNCTION udf_python_bool(x1 int, x2 int) RETURNS bool IMMUTABLE
AS $$
  return x1 > x2
$$ LANGUAGE plpythonu;
```

或者，使用 SQL 建立功能類似的 UDF，您可以使用此功能與第一個結果做比較。

```
CREATE OR REPLACE FUNCTION udf_sql_bool(int, int) RETURNS bool IMMUTABLE
AS $$
  select $1 > $2;
$$ LANGUAGE SQL;
```

3. 建立可從所建立資料表中選取並參照 UDF 的具體化視觀表。

```
CREATE MATERIALIZED VIEW mv_python_udf AS SELECT udf_python_bool(a, b) AS a FROM
 base_table;
```

您也可以選擇性地建立參照 SQL UDF 的具體化視觀表。

```
CREATE MATERIALIZED VIEW mv_sql_udf AS SELECT udf_sql_bool(a, b) AS a FROM
 base_table;
```

4. 新增資料至資料表，然後重新整理具體化視觀表。

```
INSERT INTO base_table VALUES (1,2), (1,3), (4,2);
```

```
REFRESH MATERIALIZED VIEW mv_python_udf;
```

您也可以選擇性地重新整理參照 SQL UDF 的具體化視觀表。

```
REFRESH MATERIALIZED VIEW mv_sql_udf;
```

5. 查詢具體化視觀表的資料。

```
SELECT * FROM mv_python_udf ORDER BY a;
```

查詢結果如下：

```
a
-----
false
false
true
```

這會針對最後一組值傳回 `true`，因為資料欄 a (4) 的值大於資料欄 b (2) 的值。

6. 您也可以選擇性地查詢參照 SQL UDF 的具體化視觀表。SQL 函數的結果會與 Python 版本的結果相符。

```
SELECT * FROM mv_sql_udf ORDER BY a;
```

查詢結果如下：

```
a
-----
false
false
true
```

這將針對最後一組值傳回 `true` 以進行比較。

7. 搭配 CASCADE 使用 DROP 陳述式，以捨棄使用者定義函數及參照該函數的具體化視觀表。

```
DROP FUNCTION udf_python_bool(int, int) CASCADE;
```

```
DROP FUNCTION udf_sql_bool(int, int) CASCADE;
```

串流擷取

串流擷取可從 Amazon Kinesis Data Streams 和 Amazon Managed Streaming for Apache Kafka 提供低延遲、高速擷取的串流資料至由 Amazon Redshift 佈建或 Amazon Redshift Serverless 的具體化視觀表。其可以減少存取資料所需的時間，並降低儲存成本。您可以為 Amazon Redshift 叢集或 Amazon Redshift Serverless 設定串流擷取，並使用 SQL 陳述式建立具體化視觀表，如 在 Amazon Redshift 中建立具體化視觀表 中所述。之後，使用具體化視觀表重新整理，您可以每秒擷取數百 MB 的資料。如此可快速存取快速重新整理的外部資料。

資料流程

Amazon Redshift 佈建的叢集或 Amazon Redshift Serverless 工作群組是串流取用者。具體化視觀表是從串流讀取資料的登陸區域，會在資料到達時進行處理。例如，您可以使用熟悉的 SQL 取用 JSON 值，並對應至具體化視觀表的資料欄。重新整理具體化視觀表時，Redshift 會耗用已配置 Kinesis 資料碎片或 Kafka 分割區的資料，直到檢視達到 Kinesis 串流的同位檢查，或 Kafka 主題最後一個SEQUENCE_NUMBER為止。`Offset`後續具體化視觀表會重新整理上次重新整理的最後一個 `SEQUENCE_NUMBER` 的讀取資料，直到達到串流或主題資料的同位處理為止。

串流擷取使用案例

Amazon Redshift 串流擷取的使用案例包含處理持續產生 (串流) 的資料，而且必須在產生的短時間 (延遲) 內處理。這稱為近乎即時的分析。資料來源可能有所不同，包括 IoT 裝置、系統遙測資料或來自忙碌網站或應用程式的點擊流資料。

串流擷取考量

以下是您設定串流擷取環境時，有關效能和計費的重要考量事項和最佳做法。

- 自動重新整理使用狀況和啟動 - 具體化視觀表或檢視的自動重新整理查詢會視為任何其他使用者工作負載 自動重新整理會在資料到達時從串流載入資料。

  您可以針對為串流擷取建立的具體化視觀表明確開啟自動重新整理。若要這樣做，請在具體化視觀表定義中指定 `AUTO REFRESH`。預設為手動重新整理。若要為現有具體化視觀表指定自動重新整理以進行串流擷取，您可以執行 `ALTER MATERIALIZED VIEW` 以將其開啟。如需詳細資訊，請參閱 CREATE MATERIALIZED VIEW 或 ALTER MATERIALIZED VIEW。
- 串流擷取和 Amazon Redshift Serverless - 適用於已佈建叢集上 Amazon Redshift 串流擷取的相同設定和組態指示也適用於 Amazon Redshift Serverless 上的串流擷取。使用必要的 RPU 層級調整 Amazon Redshift Serverless 的大小很重要，可支援具有自動重新整理和其他工作負載的串流擷取。如需詳細資訊，請參閱 Amazon Redshift Serverless 的計費。
- Amazon Redshift 節點位於與 Amazon MSK 叢集不同的可用區域中 - 當您設定串流擷取時，如果 Amazon MSK 啟用了機架意識，Amazon Redshift 會嘗試連線到同一可用區域中的 Amazon MSK 叢集。如果所有節點都與 Amazon Redshift 叢集位於不同的可用區域，則可能會產生跨可用區域的資料傳輸費用。為避免這種情況，請將至少一個 Amazon MSK 代理程式叢集節點保留在與 Redshift 佈建的叢集或工作群組相同的可用區域中。
- 重新整理開始位置 - 建立具體化視觀表之後，其初始重新整理會從 Kinesis 串流的 `TRIM_HORIZON` 開始，或從 Amazon MSK 主題的位移 0 開始。
- 資料格式 - 支援的資料格式僅限於可以從 `VARBYTE` 轉換的格式。如需詳細資訊，請參閱 VARBYTE 類型 及 VARBYTE 運算子。
- 將記錄附加至資料表 - 您可以執行 `ALTER TABLE APPEND` 將資料列從現有的來源具體化視觀表附加至目標資料表。這只有在具體化視觀表設定為串流擷取時才有作用。如需詳細資訊，請參閱 ALTER TABLE APPEND。
- 執行 TRUNCATE 或 DELETE - 您可以使用下列幾種方法，從用於串流擷取的具體化視觀表中移除記錄：

- TRUNCATE – 此命令會從針對串流擷取設定的具體化視觀表中刪除所有資料列。其不會執行資料表掃描。如需詳細資訊，請參閱 TRUNCATE。
- DELETE – 此命令會從針對串流擷取設定的具體化視觀表中刪除所有資料列。如需詳細資訊，請參閱 DELETE。

串流擷取最佳做法和建議

在某些情況下，您會看到如何設定串流擷取的選項。我們建議採用下列最佳作法。這些基於我們自己的測試，並通過幫助客戶避免導致數據丟失的問題。

- 從串流資料擷取值 — 如果您在具體化視觀表定義中使用 JSON_EXTRACT_PATH_TEXT 函數粉碎傳入的串流 JSON，可能會大幅影響效能和延遲。為了解釋，對於使用 JSON_EXTRACT_PATH_TEXT 提取的每個列，傳入的 JSON 被重新解析。之後，會發生任何資料類型轉換、篩選和商務邏輯。這表示，例如，如果您從 JSON 資料擷取 10 個資料欄，則會剖析每筆 JSON 記錄 10 次，其中包括類型轉換和其他邏輯。這會導致更高的擷取延遲。我們建議的另一種方法是使用 JSON_PARSE 函數將 JSON 記錄轉換為紅移的超級數據類型。串流資料登陸具體化視觀表後，請使用 PartiQL 從 SUPER 對 JSON 資料的表示擷取個別字串。如需詳細資訊，請參閱查詢半結構化資料。

  同樣重要的是要注意，JSON_EXTRACT _ 路徑 _ 文本具有最大 64 KB 的數據大小。因此，如果任何 JSON 記錄大於 64 KB，則使用 JSON_ 萃取 _ 路徑 _ 文字處理會導致錯誤。
- 將 Amazon Kinesis Data Streams 串流或 Amazon MSK 主題對應至 Amazon Redshift 串流擷取具體化視觀表 — 我們不建議建立多個串流擷取具體化視圖來擷取單一串流或 Amazon MSK 主題的資料。 Amazon Kinesis Data Streams 這是因為每個具體化視觀表會在 Kafka 主題中為 Kinesis Data Streams 或分割區中的每個碎片建立一個用戶。這可能會導致節流或超過串流或主題的輸送量。由於您多次獲取相同的數據，因此它還可能導致更高的成本。建議您為每個串流或主題建立一個串流具體化視觀表。

  如果您的使用案例要求將一個 KDS 串流或 MSK 主題的資料送入多個具體化視圖，請在執行此操作之前，先查閱AWS 大數據部落格，特別是使用 Amazon Redshift 串流擷取與 Amazon MSK 實作 near-real-time 分析的最佳實務。

使用串流擷取與 Amazon S3 中的暫存資料進行比較

將資料串流至 Amazon Redshift 或 Amazon Redshift Serverless 有多種選項。兩個眾所周知的選項是串流擷取 (如本主題所述)，或使用 Firehose 設定向 Amazon S3 的交付串流。以下清單描述了每個方法：

1. 從 Kinesis Data Streams 或 Amazon Managed Streaming for Apache Kafka 到 Amazon Redshift 或 Amazon Redshift Serverless 的串流擷取涉及設定具體化視觀表以接收資料。
2. 使用 Kinesis 資料串流將資料交付到 Amazon Redshift，並透過 Firehose 進行串流，包括將來源串流連接到亞馬遜資料火管，然後等待 Firehose 在 Amazon S3 中暫存資料。這個程序會利用不同大小的批次，並以可變長度的緩衝區作為間隔。串流到 Amazon S3 之後，Firehose 會啟動 COPY 命令來載入資料。

透過串流擷取，您可以略過第二個程序所需的幾個步驟：

- 您不必將資料傳送到 Amazon Data Firehose 交付串流，因為透過串流擷取，可以將資料直接從 Kinesis Data Streams 傳送到 Redshift 資料庫中的具體化檢視。
- 您不必在 Amazon S3 登陸串流資料，因為串流擷取資料會直接傳送至 Redshift 具體化視觀表。
- 您不需要撰寫和執行 COPY 命令，因為具體化視觀表中的資料會直接從串流重新整理。將資料從 Amazon S3 載入 Redshift 不是程序的一部分。

請注意，串流擷取僅限於來自 Amazon Kinesis Data Streams 的串流及來自 Amazon MSK 的主題。若要從 Kinesis Data Streams 至 Amazon Redshift 以外的目標，您可能需要一個 Firehose 交付串流。如需詳細資訊，請參閱將資料傳送至 Amazon 資料 Firehose 交付串流。

考量事項

以下是將擷取串流擷取至 Amazon Redshift 時的考量事項。

功能或行為	描述
Kafka 主題長度限制	Kafka 主題的名稱不得超過 128 個字元 *不包括引號)。如需詳細資訊，請參閱名稱和識別碼。
具體化視觀表的累加式重新整理和 JOIN	具體化視觀表必須是可以增量維護的。Kinesis 或 Amazon MSK 無法完整重新計算，因為預設情況下不會保留過去 24 小時或 7 天的串流或主題歷史記錄。您可以在 Kinesis 或 Amazon MSK 中設定較長的資料保留期間。但是，這可能會導致更多的維護和成本。此外，在 Kinesis 串流或 Amazon MSK 主題上建立的具體化視觀表目前不支援 JOIN。在串流或主題上建立具體化視觀表之後，您可以建立另一個具體化視觀表，將串流具體化視觀表結合至其他具體化視觀表、資料表或檢視。

功能或行為	描述
	如需詳細資訊，請參閱 REFRESH MATERIALIZED VIEW。
記錄剖析	Amazon Redshift 串流擷取不支援剖析由 inesis Producer Library 彙總的記錄 (KPL 關鍵概念 - 彙總)。彙總記錄會被擷取，但會儲存為二進制協定緩衝區資料。(如需詳細資訊，請參閱 協定緩衝區)。視您將資料推送至 Kinesis 的方式而定，您可能需要關閉此功能。
解壓縮	VARBYTE 目前不支援任何解壓縮方法。因此，包含壓縮資料的記錄無法在 Redshift 中查詢。在將資料推送到 Kinesis 串流或 Amazon MSK 主題之前，請先將資料解壓縮。
記錄大小上限	Amazon Redshift 可以從 Kinesis 或 Amazon MSK 擷取的任何記錄欄位的大小上限略小於 1MB。以下幾點詳細說明了此行為： • 最大 VARBYTE 長度 — 對於串流擷取，VARBYTE類型支援最大長度為 1,024,000 位元組的資料。Kinesis 將有效載荷限制為 1 MB。 • 訊息限制 — 預設的 Amazon MSK 組態可將訊息限制為 1 MB。此外，如果訊息包含標頭，則資料量限制為 1,048,470 個位元組。使用預設設定時，擷取沒有問題。但是，您可以將 Kafka 的訊息大小上限更改為更大的值 (Amazon MSK 也是如此)。在這種情況下，它可能是一個卡夫卡記錄的鍵/值字段，或標題，超過大小限制。這些記錄可能會導致錯誤，並且不會被擷取。 **Note** 即使 Amazon Redshift 支援資料類型的大小上限為 16 MB，Amazon Redshift 也支援從 Kinesis 或亞馬遜 MSK 進行串流擷取的最大大小為 1,024,000 位元組。VARBYTE

功能或行為	描述
錯誤記錄	在每個記錄因為資料大小超過大小限制而無法擷取至 Redshift 的情況下，就會略過該記錄。在這種情況下，具體化視觀表重新整理仍然會成功，並且每個錯誤記錄的區段都會寫入 SYS_STREAM_SCAN_ERRORS 系統表。但不會略過商務邏輯所產生的錯誤，例如計算中的錯誤或類型轉換所產生的錯誤。在將邏輯新增至具體化視觀表定義之前，請仔細測試邏輯，以避免這些問題。
Amazon MSK 多 VPC 私有連接	Redshift 串流擷取目前不支援 Amazon MSK 多 VPC 私有連線。或者，您可以使用 VPC 對等連接 VPC 或透過中央集線器連AWS Transit Gateway接 VPC 和內部部署網路。這兩種方式都可以讓 Redshift 與 Amazon MSK 叢集通訊，或與另一個 VPC 中的 Amazon MSK 無伺服器通訊。

開始使用 Amazon Kinesis Data Streams 中的串流擷取

設定 Amazon Redshift 串流擷取涉及建立對應至串流資料來源的外部結構描述，以及建立參照外部結構描述的具體化視觀表。Amazon Redshift 串流擷取支援以 Kinesis Data Streams 作為來源。因此，在設定串流擷取之前，您必須擁有可用的 Kinesis Data Streams 來源。如果您沒有來源，請按照 Kinesis 說明文件中的 Amazon Kinesis 資料串流入門中的指示進行操作，或使用透過 AWS 管理主控台建立串流中的指示在主控台上建立一個來源。

Amazon Redshift 串流擷取會使用具體化視觀表，執行 REFRESH 時會直接從串流更新該具體化視觀表。具體化視觀表會對應至串流資料來源。您可以對串流資料執行篩選和彙總，以做為具體化視觀表定義的一部分。您的串流擷取具體化視觀表 (基礎具體化視觀表) 只能參照一個串流，但是您可以建立與基礎具體化視觀表及其他具體化視觀表或資料表結合的額外具體化視觀表。

> **Note**
>
> 串流擷取和 Amazon Redshift Serverless - 本主題中的組態步驟同時適用於佈建的 Amazon Redshift 叢集和 Amazon Redshift Serverless。如需詳細資訊，請參閱 串流擷取考量。

假設您有可用的 Kinesis Data Streams 串流，第一步是使用 `CREATE EXTERNAL SCHEMA` 在 Amazon Redshift 中定義結構描述，並參照 Kinesis Data Streams 資源。之後，若要存取串流中的資料，請在具體化視觀表中定義 `STREAM`。您可以使用半結構化 `SUPER` 格式儲存串流記錄，或定義結構描述，

將資料轉換為 Redshift 資料類型。當您查詢具體化視觀表時，傳回的記錄是串流的 point-in-time 檢視表。

1. 使用信任政策來建立 IAM 角色，允許 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組擔任該角色。如需如何為 IAM 角色設定信任政策的詳細資訊，請參閱授權 Amazon Redshift 代表您存取其他 AWS 服務。建立角色之後，該角色應具有下列 IAM 政策，該政策提供與 Amazon Kinesis 資料串流通訊的許可。

   Kinesis 資料串流中未加密串流的 IAM 政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:DescribeStream"
            ],
            "Resource": "arn:aws:kinesis:*:0123456789:stream/*"
        },
        {
            "Sid": "ListStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:ListStreams",
                "kinesis:ListShards"
            ],
            "Resource": "*"
        }
    ]
}
```

   Kinesis 資料串流中加密串流的 IAM 政策

```
{
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "ReadStream",
```

```
    "Effect": "Allow",
    "Action": [
     "kinesis:DescribeStreamSummary",
     "kinesis:GetShardIterator",
     "kinesis:GetRecords",
     "kinesis:DescribeStream"
    ],
    "Resource": "arn:aws:kinesis:*:0123456789:stream/*"
   },
   {
    "Sid": "DecryptStream",
    "Effect": "Allow",
    "Action": [
     "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-
east-1:0123456789:key/1234abcd-12ab-34cd-56ef-1234567890ab"
   },
         {
    "Sid": "ListStream",
    "Effect": "Allow",
    "Action": [
     "kinesis:ListStreams",
     "kinesis:ListShards"
    ],
    "Resource": "*"
   }
  ]
}
```

2. 檢查您的 VPC，並確認您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 具有使用 NAT 閘道或網際網路閘道透過網際網路連線到 Kinesis Data Streams 端點的路徑。如果您希望 Redshift 和 Kinesis 資料串流之間的流量保留在 AWS 網路內，請考慮使用 Kinesis 介面 VPC 端點。如需詳細資訊，請參閱 使用 Amazon Kinesis Data Streams 搭配介面 VPC 端點。

3. 在 Amazon Redshift 中，建立外部結構描述以將 Kinesis 中的資料對應至結構描述。

```
CREATE EXTERNAL SCHEMA kds
FROM KINESIS
IAM_ROLE { default | 'iam-role-arn' };
```

Kinesis Data Streams 的串流擷取不需要驗證類型。它會使用 CREATE EXTERNAL SCHEMA 陳述式中定義的 IAM 角色來提出 Kinesis Data Streams 要求。

可選：使用 REGION 關鍵字來指定 Amazon Kinesis Data Streams 或 Amazon MSK 串流所在的區域。

```
CREATE EXTERNAL SCHEMA kds
FROM KINESIS
REGION 'us-west-2'
IAM_ROLE { default | 'iam-role-arn' };
```

在此範例中，該區域會指定來源串流的位置。IAM_ROLE 是一個範例。

4. 建立具體化視觀表以取用串流資料。使用類似下面的語句，如果無法解析記錄，則會導致錯誤。如果您不希望跳過錯誤記錄，請使用類似這樣的命令。

```
CREATE MATERIALIZED VIEW my_view AUTO REFRESH YES AS
SELECT *
FROM kds.my_stream_name;
```

下列範例會定義 JSON 格式來源資料的具體化視觀表。檢視會驗證傳入資料是否已正確格式化 JSON。Kinesis 串流名稱會區分大小寫，且可同時包含大寫和小寫字母。若要從具有大寫名稱的串流擷取，您可以`true`在資料庫層級`enable_case_sensitive_identifier`將組態設定為。如需詳細資訊，請參閱名稱和識別碼與 enable_case_sensitive_identifier。

```
CREATE MATERIALIZED VIEW my_view AUTO REFRESH YES AS
SELECT approximate_arrival_timestamp,
partition_key,
shard_id,
sequence_number,
refresh_time,
JSON_PARSE(kinesis_data) as kinesis_data
FROM kds.my_stream_name
WHERE CAN_JSON_PARSE(kinesis_data);
```

若要開啟自動重新整理，請使用 AUTO REFRESH YES。預設行為是手動重新整理。請注意，當您使用 CAN_JSON_PARSE 時，可能會跳過無法剖析的記錄。

中繼資料資料欄包括下列項目：

中繼資料資料欄	資料類型	描述
approximate_arrival_timestamp	沒有時區的時間戳記	將記錄插入 Kinesis 串流的大約時間
partition_key	varchar(256)	Kinesis 用來將記錄指派給碎片的索引鍵
shard_id	char(20)	從中擷取記錄的串流內的碎片唯一識別碼
sequence_number	varchar(128)	Kinesis 碎片中記錄的唯一識別碼
refresh_time	沒有時區的時間戳記	重新整理的開始時間
kinesis_data	varbyte	Kinesis 串流的記錄

請務必注意，如果您的具體化視圖定義中有業務邏輯，企業邏輯錯誤可能會導致串流擷取在某些情況下遭到封鎖。這可能會導致您必須刪除並重新建立具體化視觀表。為了避免這種情況，我們建議您盡可能簡化邏輯，並在擷取資料後對資料執行大部分的業務邏輯檢查。

5. 重新整理檢視，這會調用 Redshift 從串流讀取，並將資料載入具體化視觀表。

```
REFRESH MATERIALIZED VIEW my_view;
```

6. 查詢具體化視觀表中的資料。

```
select * from my_view;
```

開始使用 Amazon Managed Streaming for Apache Kafka 中的串流擷取

Amazon Redshift 串流擷取的目的是簡化直接從串流服務擷取串流資料至 Amazon Redshift 或 Amazon Redshift Serverless 的程序。這適用於 Amazon MSK 和 Amazon MSK Serverless 以及 Kinesis。Amazon Redshift 串流擷取不需要在將串流擷取到 Redshift 之前，在 Amazon S3 中暫存 Kinesis Data Streams 串流或 Amazon MSK 主題。

在技術層級上，串流擷取 (來自 Amazon Kinesis Data Streams 和 Amazon Managed Streaming for Apache Kafka) 可提供低延遲、高速擷取的串流或主題資料至 Amazon Redshift 具體化視觀表。在設定之後，您可以使用具體化視觀表重新整理採用大量資料。

執行下列步驟，為 Amazon MSK 設定 Amazon Redshift 串流擷取：

1. 建立對應至串流資料來源的外部結構描述。
2. 建立參照外部結構描述的具體化視觀表。

在設定 Amazon Redshift 串流擷取之前，您必須擁有可用的 Amazon MSK 來源。如果您沒有來源，請按照開始使用 Amazon MSK 中的說明進行操作。

> **Note**
>
> 串流擷取和 Amazon Redshift Serverless - 本主題中的組態步驟同時適用於佈建的 Amazon Redshift 叢集和 Amazon Redshift Serverless。如需詳細資訊，請參閱 串流擷取考量。

設定 IAM 並從 Kafka 執行串流擷取

假設您有可用的 Amazon MSK 叢集，第一步是使用 `CREATE EXTERNAL SCHEMA` 在 Redshift 中定義結構描述，並將 Kafka 主題參照為資料來源。之後，若要存取主題中的資料，請在具體化視觀表中定義 `STREAM`。您可以使用半結構化 `SUPER` 格式儲存主題中的記錄，或定義結構描述，將資料轉換為 Amazon Redshift 資料類型。當您查詢具體化視觀表時，傳回的記錄是主題的 point-in-time 檢視表。

1. 使用信任政策來建立 IAM 角色，允許 Amazon Redshift 叢集或 Amazon Redshift Serverless 擔任該角色。如需如何為 IAM 角色設定信任政策的詳細資訊，請參閱授權 Amazon Redshift 代表您存取其他 AWS 服務。建立角色之後，該角色應具有下列 IAM 政策，該政策提供與 Amazon MSK 叢集通訊的許可。如果您使用 Amazon MSK，則需要的政策取決於叢集上使用的身份驗證方法。有關 Amazon MSK 中可用的身份驗證方法，請參閱 Apache Kafka API 的身份驗證和授權。

   Amazon MSK 使用未經驗證存取的 IAM 政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": [
                "kafka:GetBootstrapBrokers"
            ],
            "Resource": "*"
        }
    ]
}
```

使用 IAM 身份驗證時適用於 Amazon MSK 的 IAM 政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MSKIAMpolicy",
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:ReadData",
                "kafka-cluster:DescribeTopic",
                "kafka-cluster:Connect"
            ],
            "Resource": [
                "arn:aws:kafka:*:0123456789:cluster/MyTestCluster/*",
                "arn:aws:kafka:*:0123456789:topic/MyTestCluster/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kafka-cluster:AlterGroup",
                "kafka-cluster:DescribeGroup"
            ],
            "Resource": [
                "arn:aws:kafka:*:0123456789:group/MyTestCluster/*"
            ]
        },
        {
            "Sid": "MSKPolicy",
            "Effect": "Allow",
            "Action": [
                "kafka:GetBootstrapBrokers"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

2. 檢查您的 VPC，並確認您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 具有前往您 Amazon MSK 叢集的路徑。Amazon MSK 叢集的輸入安全群組規則應允許您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組的安全群組。當您使用 Amazon MSK 時，您指定的連接埠取決於叢集上使用的身份驗證方法。如需詳細資訊，請參閱連接埠資訊和從 VPC 內 AWS 外存取。

   請注意，串流擷取不支援使用 MTL 進行用戶端驗證。如需詳細資訊，請參閱限制。

   下表顯示可從 Amazon MSK 設定串流擷取的免費組態選項：

Amazon Redshift 組態	Amazon MSK 組態	在 Redshift 和 Amazon MSK 之間開啟的連接埠
AUTHENTICATION NONE	TLS 傳輸已停用	9092
AUTHENTICATION NONE	已啟用的 TLS 傳輸	9094
AUTHENTICATION IAM	IAM	9098/9198

   Amazon Redshift 身份驗證是在 CREATE EXTERNAL SCHEMA 陳述式中設定的。

   如果 Amazon MSK 叢集已啟用相互傳輸層安全性 (mTLS) 身份驗證，則將 Amazon Redshift 設定為使用 AUTHENTICATION NONE 會指示其使用連接埠 9094 進行未驗證的存取。但是，這將會失敗，因為 mTLS 身份驗證正在使用連接埠。因此，我們建議您在使用 mTLS 時切換到 AUTHENTICATION IAM。

3. 在您的 Amazon Redshift 叢集或Amazon Redshift Serverless 工作群組上啟用增強型 VPC 路由。如需詳細資訊，請參閱啟用增強型 VPC 路由。

> **Note**
>
> 為了擷取 Amazon MSK 啟動程序代理程式 URL，Amazon Redshift 會使用附加的 IAM 角色提供的許可來進行GetBootstrap代理程式 API 呼叫。請注意，若要在啟用增強型 VPC 路由時才能成功執行此請求，Amazon Redshift 佈建叢集或 Amazon Redshift 無伺服器工作群組的子網路必須具有 NAT 閘道或網際網路閘道。上述子網路的網路 ACL 和安全群

組輸出規則也必須允許存取 Amazon MSK API 服務端點。如需詳細資訊，請參閱適用於 Apache Kafka 端點和配額的 Amazon 受管串流。

4. 在 Amazon Redshift 中，建立外部結構描述以對應至 Amazon MSK 叢集。

```
CREATE EXTERNAL SCHEMA MySchema
FROM MSK
IAM_ROLE { default | 'iam-role-arn' }
AUTHENTICATION { none | iam }
CLUSTER_ARN 'msk-cluster-arn';
```

在 FROM 子句中，Amazon MSK 表示結構描述會對應來自受管 Kafka 服務的資料。

當您建立外部結構描述時，Amazon MSK 的串流擷取會提供下列身份驗證類型：

- none — 指定沒有驗證步驟。
- iam — 指定 IAM 身份驗證。選擇此選項時，請確保 IAM 角色具有 IAM 身份驗證的許可。

串流擷取不支援其他 Amazon MSK 身份驗證方法，例如 TLS 驗證或使用者名稱和密碼。

CLUSTER_ARN 會指定您要從中進行串流處理的 Amazon MSK 叢集。

5. 建立具體化視觀表以取用主題的資料。如果您不想略過錯誤記錄，請使用類似此範例的 SQL 命令。

```
CREATE MATERIALIZED VIEW MyView AUTO REFRESH YES AS
SELECT *
FROM MySchema."mytopic";
```

下列範例會定義具有 JSON 來源資料的具體化視觀表。請注意，下列檢視會驗證資料是有效的 JSON 和 utf8。Kafka 主題名稱會區分大小寫，且可同時包含大寫和小寫字母。若要從具有大寫名稱的主題擷取，您可以true在資料庫層級enable_case_sensitive_identifier將組態設定為。如需詳細資訊，請參閱名稱和識別碼與 enable_case_sensitive_identifier。

```
CREATE MATERIALIZED VIEW MyView AUTO REFRESH YES AS
SELECT kafka_partition,
 kafka_offset,
 kafka_timestamp_type,
 kafka_timestamp,
 kafka_key,
```

```
 JSON_PARSE(kafka_value) as kafka_data,
 kafka_headers,
 refresh_time
FROM MySchema."mytopic"
WHERE CAN_JSON_PARSE(kafka_value);
```

若要開啟自動重新整理，請使用 AUTO REFRESH YES。預設行為是手動重新整理。

中繼資料資料欄包括下列項目：

中繼資料資料欄	資料類型	描述
kafka_partition	bigint	從 Kafka 主題記錄的分割區 ID
kafka_offset	bigint	Kafka 主題中給定分割區記錄的位移
kafka_timestamp_type	char(1)	在 Kafka 記錄中使用的時間戳記類型： • C - 客戶端上的記錄建立時間 (CREATE_TIME) • L - Kafka 服務端上的記錄附加時間 (LOG_APPEND_TIME) • U – 記錄建立時間不可用 (NO_TIMESTAMP_TYPE)
kafka_timestamp	沒有時區的時間戳記	記錄的時間戳記值
kafka_key	varbyte	Kafka 記錄的索引鍵
kafka_value	varbyte	從 Kafka 收到的記錄
kafka_headers	super	從 Kafka 收到的記錄標頭
refresh_time	沒有時區的時間戳記	重新整理的開始時間

請務必注意，如果您的具體化視圖定義中有業務邏輯，在某些情況下，商務邏輯錯誤可能會導致串流擷取區塊。這可能會導致您必須刪除並重新建立具體化視觀表。為了避免這種情況，我們建議您保持簡單的業務邏輯，並在擷取資料之後對其他資料執行其他邏輯。

6. 重新整理檢視，這會調用 Amazon Redshift 從主題讀取，並將資料載入具體化視觀表。

```
REFRESH MATERIALIZED VIEW MyView;
```

7. 查詢具體化視觀表中的資料。

```
select * from MyView;
```

執行 REFRESH 時，具體化視觀表會直接從主題更新。您可以建立對應至 Kafka 主題資料來源的具體化視觀表。您可以對資料執行篩選和彙總，以做為具體化視觀表定義的一部分。您的串流擷取具體化視觀表 (基礎具體化視觀表) 只能參照一個 Kafka 主題，但是您可以建立與基礎具體化視觀表及其他具體化視觀表或資料表結合的額外具體化視觀表。

如需串流擷取限制的相關資訊，請參閱 考量事項。

電動汽車站的資料串流擷取教學 (使用 Kinesis)

此程序會示範如何從名為 ev_station_data 的 Kinesis 串流中擷取資料，該串流包含來自不同 EV 充電站的取用資料 (採用 JSON 格式)。該結構描述是明確定義的。此範例會示範如何將資料儲存為原始 JSON，以及如何在擷取資料時將 JSON 資料轉換為 Amazon Redshift 資料類型。

生產者設定

1. 使用 Amazon Kinesis Data Streams，按照以下步驟建立名為 ev_station_data 的串流。選擇隨需做為容量模式。如需詳細資訊，請參閱透過 AWS 管理主控台建立串流。
2. Amazon Kinesis 資料產生器可協助您產生用於串流的測試資料。請依照工具中的詳細步驟來開始使用，並使用下列資料範本來產生資料：

```
{

   "_id" : "{{random.uuid}}",
   "clusterID": "{{random.number(
        {   "min":1,
            "max":50
```

```
        }
    )}}",
    "connectionTime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "kWhDelivered": "{{commerce.price}}",
    "stationID": "{{random.number(
        {   "min":1,
            "max":467
        }
    )}}",
      "spaceID": "{{random.word}}-{{random.number(
        {   "min":1,
            "max":20
        }
    )}}",

   "timezone": "America/Los_Angeles",
   "userID": "{{random.number(
        {   "min":1000,
            "max":500000
        }
    )}}"
}
```

串流資料中的每個 JSON 物件都包含下列屬性：

```
{
    "_id": "12084f2f-fc41-41fb-a218-8cc1ac6146eb",
    "clusterID": "49",
    "connectionTime": "2022-01-31 13:17:15",
    "kWhDelivered": "74.00",
    "stationID": "421",
    "spaceID": "technologies-2",
    "timezone": "America/Los_Angeles",
    "userID": "482329"
}
```

Amazon Redshift 設定

這些步驟會說明如何設定具體化視觀表以擷取資料。

1. 建立外部結構描述以將 Kinesis 中的資料對應至 Redshift 物件。

```
CREATE EXTERNAL SCHEMA evdata FROM KINESIS
IAM_ROLE 'arn:aws:iam::0123456789:role/redshift-streaming-role';
```

如需如何設定 IAM 角色的詳細資訊，請參閱 開始使用 Amazon Kinesis Data Streams 中的串流擷取。

2. 建立具體化視觀表以取用串流資料。下列範例說明兩種定義具體化視觀表以擷取 JSON 來源資料的方法。

   首先，以半結構化 SUPER 格式儲存串流記錄。在此範例中，JSON 來源會儲存在 Redshift 中，而不會轉換為 Redshift 類型。

```
CREATE MATERIALIZED VIEW ev_station_data AS
    SELECT approximate_arrival_timestamp,
    partition_key,
    shard_id,
    sequence_number,
    json_parse(kinesis_data) as payload
    FROM evdata."ev_station_data" WHERE can_json_parse(kinesis_data);
```

   相反地，在下列具體化視觀表定義中，具體化視觀表在 Redshift 中有定義的結構描述。具體化視觀表會分散在串流中的 UUID 值上，並依 approximatearrivaltimestamp 值排序。

```
CREATE MATERIALIZED VIEW ev_station_data_extract DISTKEY(6) sortkey(1) AUTO REFRESH
 YES AS
    SELECT refresh_time,
    approximate_arrival_timestamp,
    partition_key,
    shard_id,
    sequence_number,

 json_extract_path_text(from_varbyte(kinesis_data,'utf-8'),'_id',true)::character(36)
 as ID,

 json_extract_path_text(from_varbyte(kinesis_data,'utf-8'),'clusterID',true)::varchar(30)
 as clusterID,

 json_extract_path_text(from_varbyte(kinesis_data,'utf-8'),'connectionTime',true)::varchar(
 as connectionTime,
```

```
json_extract_path_text(from_varbyte(kinesis_data,'utf-8'),'kWhDelivered',true)::DECIMAL(10
as kWhDelivered,

json_extract_path_text(from_varbyte(kinesis_data,'utf-8'),'stationID',true)::DECIMAL(10,2)
as stationID,

json_extract_path_text(from_varbyte(kinesis_data,'utf-8'),'spaceID',true)::varchar(100)
as spaceID,
   json_extract_path_text(from_varbyte(kinesis_data,
'utf-8'),'timezone',true)::varchar(30)as timezone,

json_extract_path_text(from_varbyte(kinesis_data,'utf-8'),'userID',true)::varchar(30)
as userID
   FROM evdata."ev_station_data"
   WHERE LENGTH(kinesis_data) < 65355;
```

查詢串流

1. 查詢重新整理的具體化視觀表以取得使用狀況統計資料。

```
SELECT to_timestamp(connectionTime, 'YYYY-MM-DD HH24:MI:SS') as connectiontime
,SUM(kWhDelivered) AS Energy_Consumed
,count(distinct userID) AS #Users
from ev_station_data_extract
group by  to_timestamp(connectionTime, 'YYYY-MM-DD HH24:MI:SS')
order by 1 desc;
```

2. 檢視結果。

```
connectiontime         energy_consumed #users
2022-02-08 16:07:21+00   4139         10
2022-02-08 16:07:20+00   5571         10
2022-02-08 16:07:19+00   8697         20
2022-02-08 16:07:18+00   4408         10
2022-02-08 16:07:17+00   4257         10
2022-02-08 16:07:16+00   6861         10
2022-02-08 16:07:15+00   5643         10
2022-02-08 16:07:14+00   3677         10
2022-02-08 16:07:13+00   4673         10
2022-02-08 16:07:11+00   9689         20
```

在 AWS Glue Data Catalog 中建立檢視 (預覽)

這是適用於 Amazon Redshift 的資料目錄中的發行前版本文件檢視，屬於預覽版本。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。 AWS 區域 會列出目前帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的建立叢集。
5. 選擇建立叢集按鈕以建立預覽叢集。

   > **Note**
   >
   > `preview_2023` 軌跡是最近可用的預覽軌跡。此軌跡僅支援使用 RA3 節點類型建立叢集。不支援節點類型 DC2 和任何較舊的節點類型。

6. 當您的預覽叢集可用時，請使用 SQL 用戶端載入和查詢資料。

預覽功能 Data Catalog 檢視僅適用於以下區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)

- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

您也可以建立預覽工作群組來測試 Data Catalog 視觀表。您無法在生產環境中使用這些功能，也無法將工作群組移至另一個工作群組。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。如需如何建立預覽工作群組的指示，請參閱建立預覽工作群組。

透過在中建立檢視 AWS Glue Data Catalog，您可以建立單一通用檢視結構描述和中繼資料物件，以跨引擎使用，例如 Amazon Athena 和 Amazon EMR Spark。這樣做可讓您跨資料湖和資料倉儲使用相同的視觀表，以符合您的使用案例。Data Catalog 中的視觀表的特殊之處在於它們被歸類為定義者視觀表，其中存取權限由建立視觀表的使用者定義，而不是由查詢視觀表的使用者定義。以下是在 Data Catalog 中建立視觀表的一些使用案例和好處：

- 建立根據使用者需要的權限來限制資料存取的視觀表。例如，您可使用 Data Catalog 中的視觀表阻止不在 HR 部門工作的員工查看個人身分識別資訊 (PII)。
- 請確定使用者無法存取不完整的記錄。透過將某些篩選條件套用至 Data Catalog 視觀表，您可確保 Data Catalog 視觀表中的資料記錄始終是完整的。
- Data Catalog 視觀表具有包含的安全優勢，可確保用於建立視觀表的查詢定義必須完成才能建立視觀表。此安全優勢表示 Data Catalog 中的視觀表不容易受到惡意播放程式的 SQL 命令影響。
- Data Catalog 中的視觀表支援與一般視觀表相同的優點，例如允許使用者存取視觀表，而無需將基礎資料表提供給使用者。

若要在 Data Catalog 中建立視觀表，您必須具有 Spectrum 外部資料表、Lake Formation 受管資料共用中包含的物件，或 Apache Iceberg 資料表。

Data Catalog 視觀表的定義存放在 AWS Glue Data Catalog中。用 AWS Lake Formation 於透過資源授與、欄授與或以標籤為基礎的存取控制來授與存取權。如需有關在 Lake Formation 中授予和撤銷存取權的詳細資訊，請參閱授予和撤銷 Data Catalog 資源的權限。

必要條件

在 Data Catalog 中建立視觀表之前，請確定您已完成以下先決條件：

- 確定您的 IAM 角色已定義下列信任政策。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
               "Service": [
                    "glue.amazonaws.com",
                    "lakeformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- 您也需要下列傳遞角色政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com",
                        "lakeformation.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

- 最後，您也需要下列權限。
  - Glue:GetDatabase
  - Glue:GetDatabases

- Glue:CreateTable
- Glue:GetTable
- Glue:UpdateTable
- Glue:DeleteTable
- Glue:GetTables
- Glue:SearchTables
- Glue:BatchGetPartition
- Glue:GetPartitions
- Glue:GetPartition
- Glue:GetTableVersion
- Glue:GetTableVersions

電子nd-to-end 示例

首先，根據您的 Data Catalog 資料庫建立外部結構描述。

```
CREATE EXTERNAL SCHEMA IF NOT EXISTS external_schema FROM DATA CATALOG DATABASE
 'external_data_catalog_db'
IAM_ROLE 'arn:aws:iam::123456789012:role/sample-role';
```

您現在可以建立 Data Catalog 視觀表。

```
CREATE EXTERNAL PROTECTED VIEW external_schema.remote_view
AS SELECT * FROM external_schema.remote_table;
```

然後，您可以開始查詢您的視觀表。

```
SELECT * FROM external_schema.remote_view;
```

若要取得有關與 Data Catalog 中的視觀表相關之 SQL 命令的更多資訊，請參閱CREATE EXTERNAL VIEW、ALTER EXTERNAL VIEW 和 DROP EXTERNAL VIEW。

考量與限制

以下是適用於在 Data Catalog 中建立的視觀表的考量事項和限制。

- 您無法建立以其他視觀表為基礎的 Data Catalog 視觀表。
- 在 Data Catalog 視觀表中，您只能有 10 個基本資料表。
- 視觀表的定義者必須擁有基本資料表的完整 SELECT GRANTABLE 權限。
- 視觀表只能包含 Lake Formation 物件和內建項目。視觀表內不允許使用以下物件。
  - 系統表
  - 使用者定義的函數 (UDF)
  - 不在 Lake Formation 受管資料共用中的 Redshift 資料表、視觀表、具體化視觀表和後期繫結視觀表。
- 視觀表不能包含巢狀 Redshift Spectrum 表。
- 您只能使用雙點表示法查詢檢視。不支援從外部裝載的資料庫查詢 Lake Formationviews。
- Redshift 檢視中參照的 Lake Formation 資料表的 ARN 長度必須少於 127 個字元。
- AWS Glue 視圖基準物件的表現法必須與視圖位於相同 AWS 帳戶 且區域中。

在 Amazon Redshift 中查詢空間資料

空間資料描述已定義空間 (空間參考系統) 中幾何圖形的位置和形狀。Amazon Redshift 支援資料類型為 GEOMETRY 和 GEOGRAPHY 的空間資料，其中包含空間資料及資料的空間參考系統識別碼 (SRID) (選擇性)。

空間資料包含能用來表示地理特徵的幾何資料。這類資料的例子包括天氣報告、地圖方向、含地理位置的推文、商店位置，以及飛機航線。空間資料在商業分析、報告和預測方面扮演重要角色。

您可以使用 Amazon Redshift SQL 函數查詢空間資料。空間資料包含物件的幾何值。

GEOMETRY 資料類型操作會在笛卡爾平面上運作。雖然空間參照系統識別碼 (SRID) 儲存在物件內，但此 SRID 只是座標系統的識別碼，在用來處理 GEOMETRY 物件的演算法中沒有任何作用。相反地，GEOGRAPHY 資料類型上的操作會將物件內部的座標視為球體上的球形座標。此球體由參考地理空間參考系統的 SRID 定義。依預設，GEOGRAPHY 資料類型是使用空間參考 (SRID) 4326 建立的，並且參考世界大地測量系統 (WGS) 84。如需 SRID 的相關資訊，請參閱 Wikipedia 中的空間參考系統。

您可以使用 ST_Transform 函數來轉換來自各種空間參考系統的座標。坐標轉換完成後，只要輸入 GEOMETRY 使用地理 SRID 編碼，您也可以在兩者之間使用簡單的轉換。這種轉換只是複製坐標而無需進一步的轉換。例如：

```
SELECT ST_AsEWKT(ST_GeomFromEWKT('SRID=4326;POINT(10 20)')::geography);
```

```
st_asewkt
------------------------
 SRID=4326;POINT(10 20)
```

為了更好地了解 GEOMETRY 和 GEOGRAPHY 資料類型之間的差異，可考慮使用世界大地測量系統 (WGS) 84 來計算柏林機場 (BER) 和舊金山機場 (SFO) 之間的距離。使用 GEOGRAPHY 資料類型時，結果會以公尺為單位。將 GEOMETRY 資料類型與 SRID 4326 搭配使用時，結果會以度為單位，無法轉換為公尺，因為一度的距離取決於地球幾何圖形上的位置。

對 GEOGRAPHY 資料類型的計算主要用於現實的圓形地球計算，如一個國家/地區精確而不失真的面積。但是其計算成本要高得多。因此，ST_Transform 可以將您的座標轉換為適當的局部投影座標系統，並更快地對 GEOMETRY 資料類型進行計算。

使用空間資料，您可以執行查詢來執行以下作業：

- 尋找兩點之間的距離。

- 檢查是否一個區域 (多邊形) 包含其他區域。
- 檢查一個線串 (linestring) 是否與另一個線串或多邊形交集。

您可以使用 GEOMETRY 資料類型來保存空間資料的值。Amazon Redshift 中的 GEOMETRY 值可以定義二維 (2D)、三維 (3DZ)、具量值二維 (3DM) 及四維 (4D) 幾何基本資料類型：

- 二維 (2D) 幾何圖形由平面中的兩個直角座標 (x、y) 指定。
- 三維 (3DZ) 幾何圖形由空間中的三個直角座標 (x、y、z) 指定。
- 具量值二維 (3DM) 幾何圖形由三個坐標 (x、y、m) 指定，前兩個座標是平面中的直角坐標，第三個是測量值。
- 四維 (4D) 幾何圖形由四個坐標 (x、y、z、m) 指定，前三個座標是空間中的直角坐標，第四個是測量值。

如需基本幾何資料類型的相關資訊，請參閱 Wikipedia 中的 Well-known text representation of geometry (幾何圖形的熟知文字表示法)。

您可以使用 GEOGRAPHY 資料類型來保存空間資料的值。Amazon Redshift 中的 GEOGRAPHY 值可以定義二維 (2D)、三維 (3DZ)、具量值二維 (3DM) 及四維 (4D) 幾何基本資料類型：

- 二維 (2D) 幾何圖形由球體上的經度和緯度座標指定。
- 三維 (3DZ) 幾何圖形由球體上的經度、緯度和高度座標指定。
- 具量值二維 (3DM) 幾何圖形由三個坐標 (經度、緯度和量值) 指定，前兩個座標是球體上的角度坐標，第三個是測量值。
- 四維 (4D) 幾何圖形由四個座標 (經度、緯度、高度、量值) 指定，其中前三個是經度、緯度和高度，第四個是測量值。

如需地理坐標系統的相關資訊，請參閱 Wikipedia 中的地理坐標系統和球形坐標系統。

GEOMETRY 和 GEOGRAPHY 資料類型具有下列子類型：

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING

- MULTIPOLYGON
- GEOMETRYCOLLECTION

其中有支援下列幾何資料表示法的 Amazon Redshift SQL 函數：

- GeoJSON
- 已知文字 (WKT)
- 擴充的已知文字 (EWKT)
- 已知二進位 (WKB) 表示法
- 擴充的已知二進位 (EWKB)

您可以在 GEOMETRY 和 GEOGRAPHY 資料類型之間進行轉換。

下列 SQL 將線串從 GEOMETRY 轉換為 GEOGRAPHY。

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)')::geography);
```

```
 st_asewkt
---------------------------------------------
 SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

下列 SQL 將線串從 GEOGRAPHY 轉換為 GEOMETRY。

```
SELECT ST_AsEWKT(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)')::geometry);
```

```
 st_asewkt
---------------------------------------------
 SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

Amazon Redshift 提供許多 SQL 函數來查詢空間資料。除了 ST_IsValid 函數之外，接受 GEOMETRY 物件作為引數的空間函數會預期此 GEOMETRY 物件是有效的幾何圖形。如果 GEOMETRY 或 GEOGRAPHY 物件無效，則不會定義空間函數的行為。如需有效性的相關資訊，請參閱 幾何有效性。

如需查詢空間資料 SQL 函數的詳細資訊，請參閱 空間函數。

如需載入空間資料的詳細資訊，請參閱 載入 GEOMETRY 或 GEOGRAPHY 資料類型的欄。

主題

教學課程：將空間 SQL 函數與 Amazon Redshift 搭配使用

本教學課程示範如何將部分空間 SQL 函數與 Amazon Redshift 搭配使用。

若要這麼做，您可以使用空間 SQL 函數來查詢兩個資料表。該教學課程會使用公共資料集中的資料，該資料將德國柏林的出租住所位置資料與郵遞區號建立關聯。

主題

必要條件

在此教學課程中，您需執行下列資源：

- 您可以存取和更新的現有 Amazon Redshift 叢集和資料庫。在現有叢集中，您可以建立資料表、載入範例資料，以及執行 SQL 查詢以示範空間函數。您的叢集至少要有兩個節點。若要了解如何建立叢集，請按照 Amazon Redshift 入門指南中的步驟進行操作。
- 若要使用 Amazon Redshift 查詢編輯器，請確定您的叢集位於支援查詢編輯器的 AWS 區域中。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的使用查詢編輯器來查詢資料庫。
- AWS 您的 Amazon Redshift 叢集的登入資料，可讓叢集從 Amazon S3 載入測試資料。如需如何存取 Amazon S3 等其他 AWS 服務的相關資訊，請參閱授權 Amazon Redshift 存取 AWS服務。
- 名為的 AWS Identity and Access Management (IAM) 角色`mySpatialDemoRole`，其中`AmazonS3ReadOnlyAccess`附加了受管政策以讀取 Amazon S3 資料。若要建立有權從

Amazon S3 儲存貯體載入資料的角色，請參閱 Amazon Redshift 管理指南中的使用 IAM 角色授權 COPY、UNLOAD 和 CREATE EXTERNAL SCHEMA 操作。

- 建立 IAM 角色 mySpatialDemoRole 之後，該角色需要與您的 Amazon Redshift 叢集建立關聯。如需如何建立該關聯的相關資訊，請參閱《Amazon Redshift 管理指南》中的使用 IAM 角色授權 COPY、UNLOAD 和 CREATE EXTERNAL SCHEMA 操作。

步驟 1：建立資料表並載入測試資料

本教學課程使用的來源資料來自名為 accommodations.csv 和 zipcodes.csv 的檔案。

accommodations.csv 檔案是來自 insideairbnb.com 的開放原始碼資料。zipcodes.csv 檔案提供的郵遞區號是來自德國柏林布蘭登堡國家統計機構 (Amt für Statistik Berlin-Brandenburg) 的開放原始碼資料。兩個資料來源皆經過創用 CC (Creative Commons) 授權。該資料僅限於德國柏林地區。這些檔案位於 Amazon S3 公用儲存貯體中，可與本教學課程搭配使用。

您可以選擇性地從下列 Amazon S3 連結下載來源資料：

- accommodations 資料表的來源資料。
- zipcode 資料表的來源資料。

使用下列程序來建立資料集並載入測試資料。

建立資料表並載入測試資料

1. 開啟 Amazon Redshift 查詢編輯器。如需使用查詢編輯器的相關資訊，請參閱《Amazon Redshift 管理指南》中的使用查詢編輯器來查詢資料庫。
2. 刪除本教學課程使用的任何資料表 (如果這些資料表已經存在於您的資料庫中)。如需詳細資訊，請參閱 步驟 3：清除您的資源。
3. 建立 accommodations 資料表以儲存每個住所的地理位置 (經度和緯度)、清單名稱以及其他業務資料。

   本教學課程會探討德國柏林的房間租賃。shape 資料欄儲存的是住所位置的地理點。其他資料欄包含有關租賃的資訊。

   若要建立 accommodations 資料表，請在 Amazon Redshift 查詢編輯器中執行下列 SQL 陳述式。

```
CREATE TABLE public.accommodations (
```

```
  id INTEGER PRIMARY KEY,
  shape GEOMETRY,
  name VARCHAR(100),
  host_name VARCHAR(100),
  neighbourhood_group VARCHAR(100),
  neighbourhood VARCHAR(100),
  room_type VARCHAR(100),
  price SMALLINT,
  minimum_nights SMALLINT,
  number_of_reviews SMALLINT,
  last_review DATE,
  reviews_per_month NUMERIC(8,2),
  calculated_host_listings_count SMALLINT,
  availability_365 SMALLINT
);
```

4. 在查詢編輯器中建立 zipcode 資料表以儲存柏林郵遞區號。

   郵遞區號在 wkb_geometry 資料欄中定義為多邊形。其餘資料欄會描述有關郵遞區號的其他空間中繼資料。

   若要建立 zipcode 資料表，請在 Amazon Redshift 查詢編輯器中執行下列 SQL 陳述式。

```
CREATE TABLE public.zipcode (
  ogc_field INTEGER PRIMARY KEY NOT NULL,
  wkb_geometry GEOMETRY,
  gml_id VARCHAR(256),
  spatial_name VARCHAR(256),
  spatial_alias VARCHAR(256),
  spatial_type VARCHAR(256)
 );
```

5. 使用範例資料來載入資料表。

   本教學的範例資料是在 Amazon S3 儲存貯體中提供，允許讀取所有經驗證的 AWS 使用者。請確定您已提供允許存取 Amazon S3 的有效 AWS 登入資料。

   若要將測試資料載入資料表，請執行下列 COPY 命令。將 *account-number* 換成您自己的 AWS 帳戶號碼。以單引號括住的登入資料字串區段不可包含任何空格或分行符號。

```
COPY public.accommodations
FROM 's3://redshift-downloads/spatial-data/accommodations.csv'
DELIMITER ';'
```

```
IGNOREHEADER 1 REGION 'us-east-1'
CREDENTIALS 'aws_iam_role=arn:aws:iam::account-number:role/mySpatialDemoRole';
```

```
COPY public.zipcode
FROM 's3://redshift-downloads/spatial-data/zipcode.csv'
DELIMITER ';'
IGNOREHEADER 1 REGION 'us-east-1'
CREDENTIALS 'aws_iam_role=arn:aws:iam::account-number:role/mySpatialDemoRole';
```

6. 執行下列命令，可驗證每一個資料表是否正確載入。

```
select count(*) from accommodations;
```

```
select count(*) from zipcode;
```

下列結果會顯示每個測試資料資料表的列數。

資料表名稱	資料列
住所	22,248
zipcode	190

步驟 2：查詢空間資料

建立並載入資料表之後，您可以使用 SQL SELECT 陳述式來查詢這些資料表。下列查詢示範的是一些您可以擷取的資訊。您可以編寫許多其他使用空間函數來滿足您需求的查詢。

查詢空間資料

1. 查詢以取得 accommodations 資料表中所儲存清單項目的總數，如下所示。空間參考系統是世界大地測量系統 (WGS) 84，具有唯一的空間參考識別碼 4326。

```
SELECT count(*) FROM public.accommodations WHERE ST_SRID(shape) = 4326;
```

```
 count
-------
```

```
 22248
```

2. 擷取格式為熟知文字 (WKT) 且具有一些其他屬性的幾何圖形物件。此外，您可以驗證郵遞區號資料是否也儲存在使用空間參考 ID (SRID) 4326 的世界大地測量系統 (WGS) 84 中。空間資料必須儲存在相同的空間參考系統中才能互通。

```
SELECT ogc_field, spatial_name, spatial_type, ST_SRID(wkb_geometry),
 ST_AsText(wkb_geometry)
FROM public.zipcode
ORDER BY spatial_name;
```

```
ogc_field  spatial_name  spatial_type  st_srid  st_astext
-----------------------------------------------------------------
0          10115        Polygon        4326     POLYGON((...))
4          10117        Polygon        4326     POLYGON((...))
8          10119        Polygon        4326     POLYGON((...))
...
(190 rows returned)
```

3. 選取柏林行政區-柏林米特區 (10117) 的多邊形 (GeoJSON 格式)、其維度和此多邊形中的點數。

```
SELECT ogc_field, spatial_name, ST_AsGeoJSON(wkb_geometry),
 ST_Dimension(wkb_geometry), ST_NPoints(wkb_geometry)
FROM public.zipcode
WHERE spatial_name='10117';
```

```
ogc_field  spatial_name  spatial_type
 st_dimension  st_npoint
-------------------------------------------------------------------------------------------
4          10117         {"type":"Polygon", "coordinates":[[[...]]]}    2
    331
```

4. 執行以下 SQL 命令以檢視布蘭登堡門周圍 500 公尺內有多少住所。

```
SELECT count(*)
FROM public.accommodations
WHERE ST_DistanceSphere(shape, ST_GeomFromText('POINT(13.377704 52.516431)', 4326))
 < 500;
```

```
count
------
  29
```

5. 透過執行以下查詢，從附近所列出住所中儲存的資料取得布蘭登堡門的粗略位置。

   此查詢需要子選取 (subselect)。這會產生不同的計數，因為要求的位置與以前的查詢不一樣，這更接近住所。

```
WITH poi(loc) as (
  SELECT st_astext(shape) FROM accommodations WHERE name LIKE '%brandenburg gate%'
)
SELECT count(*)
FROM accommodations a, poi p
WHERE ST_DistanceSphere(a.shape, ST_GeomFromText(p.loc, 4326)) < 500;
```

```
count
------
  60
```

6. 執行以下查詢以顯示布蘭登堡門周圍所有住所的詳細資訊 (按價格遞減排序)。

```
SELECT name, price, ST_AsText(shape)
FROM public.accommodations
WHERE ST_DistanceSphere(shape, ST_GeomFromText('POINT(13.377704 52.516431)', 4326))
 < 500
ORDER BY price DESC;
```

```
name                                                price  st_astext
-------------------------------------------------------------------------------------------
DUPLEX APARTMENT/PENTHOUSE in 5* LOCATION! 7583        300
 POINT(13.3826510209548 52.5159819722552)
DUPLEX-PENTHOUSE IN FIRST LOCATION! 7582              300
 POINT(13.3799997083855 52.5135918444834)
...
(29 rows returned)
```

7. 執行以下查詢，透過郵遞區號擷取最昂貴的住所。

```
SELECT
  a.price, a.name, ST_AsText(a.shape),
  z.spatial_name, ST_AsText(z.wkb_geometry)
FROM accommodations a, zipcode z
WHERE price = 9000 AND ST_Within(a.shape, z.wkb_geometry);
```

```
price   name                                    st_astext
     spatial_name      st_astext
-------------------------------------------------------------------------------------------
9000    Ueber den Dächern Berlins Zentrum    POINT(13.334436985013
 52.4979779501538)    10777             POLYGON((13.3318284987227
 52.4956021172799,...
```

8. 透過使用子查詢來計算出最高，最低或中等的住所價格。

   下列查詢依郵遞區號列出了中等的住所價格。

```
SELECT
  a.price, a.name, ST_AsText(a.shape),
  z.spatial_name, ST_AsText(z.wkb_geometry)
FROM accommodations a, zipcode z
WHERE
  ST_Within(a.shape, z.wkb_geometry) AND
  price = (SELECT median(price) FROM accommodations)
ORDER BY a.price;
```

```
price name                              st_astext
 spatial_name   st_astext
-------------------------------------------------------------------------------------------

45    "Cozy room Berlin-Mitte"        POINT(13.3864349535358 52.5292016386514)
 10115          POLYGON((13.3658598465795 52.535659581048,...
...
(723 rows returned)
```

9. 執行以下查詢以擷取柏林中列出的住所數量。為了找到熱點，這些住所會按郵遞區號進行分組，並按供應量排序。

```
SELECT z.spatial_name as zip, count(*) as numAccommodations
FROM public.accommodations a, public.zipcode z
WHERE ST_Within(a.shape, z.wkb_geometry)
GROUP BY zip
ORDER BY numAccommodations DESC;
```

```
zip   numaccommodations
---------------------------
10245 872
10247 832
10437 733
10115 664
...
(187 rows returned)
```

步驟 3：清除您的資源

叢集只要執行就會繼續產生費用。完成本教學課程後，您可以刪除範例叢集。

如果您要保留叢集，但又想復原測試資料表所使用的儲存體，請執行下列命令來刪除資料表。

```
drop table public.accommodations cascade;
```

```
drop table public.zipcode cascade;
```

將 Shapefile 載入 Amazon Redshift

您可以使用 COPY 命令將儲存在 Amazon S3 中的 Esri Shapefile 擷取到 Amazon Redshift 資料表中。Shapefile 會以向量格式儲存地理特徵的幾何位置和屬性資訊。Shapefile 格式可以在空間上描述空間物件，例如點、線和多邊形。如需 Shapefile 的相關資訊，請參閱 Wikipedia 中的 Shapefile。

COPY 命令支援資料格式參數 SHAPEFILE。依預設，Shapefile 的第一個資料欄是 GEOMETRY 或 IDENTITY 資料欄。所有後續的資料欄都會遵循 Shapefile 中指定的順序。但是，目標資料表不需要使用此確切配置，因為您可以使用 COPY 資料欄對應來定義順序。若要取得有關 COPY 命令 Shapefile 支援的資訊，請參閱 SHAPEFILE。

在某些情況下，產生的幾何圖形大小可能會大於 Amazon Redshift 中可儲存幾何圖形的最大值。如果是這樣，您可以使用 COPY 選項 SIMPLIFY 或 SIMPLIFY AUTO 簡化擷取期間的幾何圖形，如下所示：

- 指定 SIMPLIFY *tolerance* 可使用 Ramer-Douglas-Peucker 演算法和指定的容許值來簡化擷取期間的所有幾何圖形。
- 指定 SIMPLIFY AUTO 但不含容許值，可使用 Ramer-Douglas-Peucker 演算法來單獨簡化大於大小上限的幾何圖形。此方法會計算仍可在大小上限內儲存物件的最小容許值。
- 指定 SIMPLIFY AUTO *max_tolerance*，可使用 Ramer-Douglas-Peucker 演算法和自動計算的容許值來單獨簡化大於大小上限的幾何圖形。此方法可確保容許值不超過容許值上限。

如需 GEOMETRY 資料值大小上限的詳細資訊，請參閱 將空間資料與 Amazon Redshift 搭配使用時的考量事項。

在某些情況下，容許值會低到記錄無法縮小到 GEOMETRY 資料值的大小上限。在這些情況下，您可以使用 COPY 命令的 MAXERROR 選項忽略所有或最多一定數量的擷取錯誤。

COPY 命令也支援載入 GZIP Shapefile。若要執行此作業，請指定 COPY GZIP 參數。使用此選項時，必須獨立壓縮所有 Shapefile 元件，並共用相同的壓縮字尾。

如果 Shapefile 中存在投影描述檔 (.prj)，Redshift 會使用它來決定空間參考系統識別碼 (SRID)。如果 SRID 有效，則產生的幾何圖形會獲派此 SRID。如果與輸入幾何圖形相關聯的 SRID 值不存在，則產生的幾何圖形的 SRID 值為零。您可以將 SET read_srid_on_shapefile_ingestion 設定為 OFF，在工作階段層級停用空間參考系統 ID 的自動偵測。

查詢 SYS_SPATIAL_SIMPLIFY 或 SVL_SPATIAL_SIMPLIFY 系統檢視，即可檢視已簡化的記錄及已計算的容許值。當您指定 SIMPLIFY *tolerance* 時，此檢視會包含每個 COPY 操作的記錄。若不指定，則會包含每個已簡化幾何圖形的記錄。如需詳細資訊，請參閱 SYS_SPATIAL_SIMPLIFY 或 SVL_SPATIAL_SIMPLIFY 。

如需載入 Shapefile 的範例，請參閱 將 Shapefile 載入 Amazon Redshift。

Amazon Redshift 空間資料的術語

下列術語用來描述 Amazon Redshift 的一些空間函數。

邊界框

幾何或地理的週框方塊會定義為幾何或地理中所有點座標範圍的交叉乘積 (跨維度)。對於二維幾何圖形，週框方塊是完全包含幾何中所有點的矩形。例如，多邊形 POLYGON((0 0,1 0,0 2,0 0)) 的週框方塊是由點 (0, 0) 和 (1, 2) 定義為其左下角和右上角的矩形。Amazon Redshift 會在幾何圖形中預先計算並儲存週框方塊，以加速形成幾何述詞和空間聯結。例如，如果兩個幾何圖形的週框方塊不相交，則這兩個幾何圖形就不會相交，而且不能存在於使用 st_Intersect 述詞的空間聯結結果集中。

您可以使用空間函數來加入 (AddBBox)、捨棄 (DropBBox) 及決定週框方塊的支援 (SupportsBBox)。Amazon Redshift 支援對所有幾何子類型預先計算週框方塊。

下列範例顯示如何更新資料表中的現有幾何圖形，以將其與週框方塊一起儲存。如果您叢集的叢集版本為 1.0.26809 或更新版本，則依預設會使用預先計算的週框方塊建立所有新的幾何圖形。

```
UPDATE my_table SET geom = AddBBox(geom) WHERE SupportsBBox(geom) = false;
```

更新現有幾何圖形之後，我們建議您在更新的資料表格執行 VACUUM 指令。如需詳細資訊，請參閱 VACUUM。

若要設定工作階段期間是否使用週框方塊對幾何圖形進行編碼，請參閱 default_geometry_encoding。

幾何有效性

Amazon Redshift 使用的幾何演算法會假設輸入幾何圖形是有效的幾何圖形。如果演算法的輸入無效，則結果處於未定義狀態。下節說明 Amazon Redshift 針對每個幾何子類型使用的幾何有效性定義。

點 (Point)

若以下其中一個條件成立，則點將視為有效：

- 點是空點。
- 所有點座標均為有限浮點數。

點可以是空點。

線串 (Linestring)

若以下任何一個條件成立，則線串將視為有效：

- 線串是空的，也就是不包含任何點。
- 非空線串中的所有點都具有使用有限浮點數的座標。
- 如果線串不是空的，則必須是一維線條；也就是不能退化為一個點。

線串不能包含空點。

線串可以具有重複的連續點。

線串可以具有自我相交。

多邊形 (Polygon)

若以下任何一個條件成立，則多邊形將視為有效：

- 多邊形是空的，也就是不包含任何環。
- 如果不是空的，則當下列所有條件皆成立時，多邊形為有效：
  - 多邊形的所有環都是有效的。若以下任何所有條件皆成立，則環將視為有效：
    - 環的所有點都具有使用有限浮點數的座標。
    - 環是封閉的；也就是環的第一個點和最後一個點重合。
    - 環沒有任何自我交集。
    - 環是二維的。
  - 多邊形的環具有一致的方向。也就是說，如果您遍歷任何環，多邊形的內部還會在您的右側或左側。這意味著，如果多邊形的外部環是順時針或逆時針方向，則所有多邊形的內部環必須具有相同的逆時鐘或順時針方向。
  - 所有內環必須位 於多邊形的外環內。
  - 內環不能是巢狀結構，也就是說，內環不能在另一個內環內。
  - 內環和外環只能在有限數量的點上相交。
  - 多邊形的內部必須單連通。

多邊形不能包含空點。

多點 (Multipoint)

若以下任何一個條件成立，則多點將視為有效：

- 多點是空的，也就是不包含任何點。
- 多點不是空的，且根據點有效性定義，所有點都是有效的。

多點可以包含一個或多個空點。

多點可以有重複的點。

多重線串 (Multiinestring)

若以下任何一個條件成立，則多重線串將視為有效：

- 多重線串是空的，也就是不包含任何線串。
- 根據線串有效性定義，非空多重線串中的所有線串都是有效的。

僅由空線串組成的非空多重線串會視為有效。

多重線串中的空線串不會影響其有效性。

多重線串可以具有包含重複連續點的線串。

多重線串可以具有自我相交。

多重線串不能包含空點。

多重多邊形 (Multipolygon)

若以下任何一個條件成立，則多重多邊形將視為有效：

- 多重多邊形不包含任何多邊形 (也就是空的)。
- 多重多邊形不是空的，且下列所有條件皆成立：
  - 多重多邊形中的所有多邊形都是有效的。
  - 多重多邊形中沒有兩個多邊形可以在無限數量的點上相交。特別是，這意味著任何兩個多邊形的內部不能相交，並且只能在有限數量的點上碰到。

多重多邊形中的空多邊形不會使多重多邊形無效。

多重多邊形不能包含空點。

幾何集合

若以下任何一個條件成立，則幾何集合將視為有效：

- 幾何集合是空的，也就是不包含任何幾何圖形。
- 非空幾何集合中的所有幾何圖形都是有效的。

此定義仍然適用於巢狀幾何圖形集合，但是以遞迴方式套用。

幾何集合可以包含空點和具有空點的多點。

幾何簡單性

Amazon Redshift 使用的幾何演算法會假設輸入幾何圖形是有效的幾何圖形。如果演算法的輸入無效，則簡單性檢查會處於未定義狀態。下節說明 Amazon Redshift 針對每個幾何子類型使用的幾何簡單性定義。

點 (Point)

若以下任何一個條件成立，則有效點將視為簡單：

- 有效點一律視為簡單。
- 空點視為簡單。

線串 (Linestring)

若以下任何一個條件成立，則有效線串將視為簡單：

- 線串是空的。
- 線串不是空的，且下列所有條件皆成立：
  - 線串沒有重複的連續點。
  - 線串沒有自我相交，除了可能是會重合的第一個點和最後一個點。換句話說，除了在邊界點之外，線串不能具有自我相交。

多邊形

如果有效多邊形不包含任何重複的連續點，則視為簡單多邊形。

多點 (Multipoint)

若以下任何一個條件成立，則有效多點將視為簡單：

- 多點是空的，也就是不包含任何點。
- 多點的兩個非空點不重合。

多重線串 (Multilinestring)

若以下任何一個條件成立，則有效多重線串將視為簡單：

- 多重線串是空的。
- 多重線串不是空的，且下列所有條件皆成立：
  - 多重線串的所有線串都是簡單線串。
  - 多重線串的任何兩條線串都不相交，除了作為兩個線串邊界點的點。

僅由空線串組成的非空多重線串會視為空多重線串。

多重線串中的空線串不會影響其簡單性。

多重線串中的封閉線串不能與多重線串中的任何其他線串相交。

多重線串不可具有包含重複連續點的線串。

多重多邊形 (Multipolygon)

如果有效多重多邊形不包含任何重複的連續點，則視為簡單多邊形。

幾何集合

若以下任何一個條件成立，則有效幾何集合將視為簡單：

- 幾何集合是空的，也就是不包含任何幾何圖形。
- 非空幾何集合中的所有幾何圖形都是簡單的。

此定義仍然適用於巢狀幾何圖形集合，但是以遞迴方式套用。

H3

H3 是一種分層式地理空間索引網格系統，此系統會提供了一種將空間座標索引到平方公尺解析度的方法。索引的資料可以跨不同的資料集聯結，並以不同的精確程度彙總。H3 根據格點啟用一系列演算法和最佳化，包括最近鄰點、最短路徑、漸層平滑化等。H3 索引指的是可以是六邊形或五邊形的儲存格。該空間依照解析度被分層細分。H3 支援 0-15 的 16 個解析度，內含。0 是最普遍的，15 是最好。

Amazon Redshift 提供以下 H3 空間函數：

- H3_ FromLongLat
- H3_ FromPoint
- H3_Polyfill

將空間資料與 Amazon Redshift 搭配使用時的考量事項

將空間資料與 Amazon Redshift 搭配使用時的考量事項如下：

- GEOMETRY 或 GEOGRAPHY 物件的大小上限是 1,048,447 位元組。
- Amazon Redshift Spectrum 本身不支援空間資料。因此，您無法建立或改變包含 GEOMETRY 或 GEOGRAPHY 資料欄的外部資料表。
- Python 使用者定義函數 (UDF) 的資料類型不支援 GEOMETRY 或 GEOGRAPHY 資料類型。
- 您無法使用 GEOMETRY 或 GEOGRAPHY 資料欄做為 Amazon Redshift 資料表的排序索引鍵或分配索引鍵。

- 您無法在 SQL ORDER BY、GROUP BY 或 DISTINCT 子句中使用 GEOMETRY 或 GEOGRAPHY 資料欄。
- 您無法在許多 SQL 函數中使用 GEOMETRY 或 GEOGRAPHY 資料欄。
- 您無法在 GEOMETRY 或 GEOGRAPHY 資料欄上執行 UNLOAD 操作至每個格式。您可以將 GEOMETRY 或 GEOGRAPHY 資料欄卸載為文字或逗號分隔值 (CSV) 檔案。這樣做會以十六進制 EWKB 格式寫入 GEOMETRY 或 GEOGRAPHY 資料。如果 EWKB 資料的大小超過 4 MB，則會發生警告，因為資料稍後將無法載入資料表。
- GEOMETRY 或 GEOGRAPHY 資料支援的壓縮編碼是 RAW。
- 使用 JDBC 或 ODBC 驅動程式時，請使用自訂類型映射。在此案例中，用戶端應用程式必須具備資訊，指出 ResultSet 物件的哪些參數是 GEOMETRY 或 GEOGRAPHY 物件。ResultSetMetadata 操作會傳回 VARCHAR 類型。
- 若要從 SHAPEFILE 複製地理日期，請先擷取至 GEOMETRY 欄中，然後將物件轉換為 GEOGRAPHY 物件。

下列非空間函數可接受 GEOMETRY 或 GEOGRAPHY 類型的輸入，或是 GEOMETRY 或 GEOGRAPHY 類型的資料欄：

- COUNT 彙整函數
- COALESCE 和 NVL 條件表達式
- CASE 表達式
- GEOMETRY 和 GEOGRAPHY 的預設編碼為 RAW。如需更多詳細資訊，請參閱 壓縮編碼。

使用 Amazon Redshift 中的聯合查詢來查詢資料

透過在 Amazon Redshift 中使用聯合查詢，您可以跨操作資料庫、資料倉儲和資料湖查詢和分析資料。使用聯合查詢功能，您可以在外部資料庫的即時資料上將來自 Amazon Redshift 的查詢與跨 Amazon Redshift 和 Amazon S3 環境的查詢整合在一起。聯合查詢可以使用 Amazon RDS for PostgreSQL、Amazon Aurora PostgreSQL 相容版本、Amazon RDS for MySQL 和 Amazon Aurora MySQL 相容版本中的外部資料庫。

您可以使用聯合查詢，將即時資料整合為商業智慧 (BI) 和報告應用程式的一部分。例如，若要讓資料擷取至 Amazon Redshift 更容易，您可以使用聯合查詢來執行下列動作：

- 直接查詢操作資料庫。
- 快速套用變換。
- 將資料載入到目標資料表，而不需要複雜的擷取、轉換、載入 (ETL) 管道。

為了減少透過網路的資料移動並改善效能，Amazon Redshift 會將聯合查詢的部分計算直接分配到遠端操作資料庫。Amazon Redshift 也會視需要使用其平行處理能力來支援執行這些查詢。

執行聯合查詢時，Amazon Redshift 會先從領導節點與 RDS 或 Aurora 資料庫叢集資料庫執行個體建立用戶端連線，以擷取表格中繼資料。從運算節點中，Amazon Redshift 會發出述詞向下推送的子查詢並擷取結果資料列。然後，Amazon Redshift 會在運算節點之間分配結果資料列，以便進一步處理。

傳送至 Amazon Aurora PostgreSQL 資料庫或 Amazon RDS for PostgreSQL 資料庫的查詢相關詳細資訊記錄在系統檢視 SVL_FEDERATED_QUERY 中。

主題

- 開始使用 PostgreSQL 的聯合查詢
- 開始使用與 PostgreSQL 的同盟查詢 AWS CloudFormation
- 開始使用 MySQL 的聯合查詢
- 建立秘密和 IAM 角色來使用聯合查詢
- 使用聯合查詢的範例
- Amazon Redshift 與支援的 PostgreSQL 和 MySQL 資料庫之間的資料類型差異
- 使用 Amazon Redshift 存取聯合資料時的注意事項

開始使用 PostgreSQL 的聯合查詢

若要建立聯合查詢，您可以遵循以下一般方法：

1. 設定從您的 Amazon Redshift 叢集到 Amazon RDS 或 Aurora PostgreSQL DB 執行個體的連線。

   若要這樣做，請確定 RDS PostgreSQL 或 Aurora PostgreSQL DB 執行個體可以接受來自 Amazon Redshift 叢集的連線。我們建議您的 Amazon Redshift 叢集和 Amazon RDS 或 Aurora PostgreSQL 執行個體位於相同的虛擬私有雲端 (VPC) 和子網路群組中。如此一來，您就可以將 Amazon Redshift 叢集的安全群組新增至 RDS 或 Aurora PostgreSQL DB 執行個體的安全群組傳入規則。

   您也可以設定 VPC 對等，或其他允許 Amazon Redshift 與 RDS 或 Aurora PostgreSQL 執行個體建立連線的網路。如需 VPC 網路的相關資訊，請參閱以下內容。

   - 《Amazon VPC 對等互連指南》中的什麼是 VPC 對等互連？
   - 《Amazon RDS 使用者指南》中的在 VPC 中使用 DB 執行個體。

   > **Note**
   >
   > 在某些情況下，您必須啟用增強型 VPC 路由：例如，如果您的 Amazon Redshift 叢集與 RDS 或 Aurora PostgreSQL 執行個體位於不同的 VPC 中，或者它們位於相同的 VPC 中，且您的路由需要這麼做。否則，當您執行聯合查詢時，您可能會收到逾時錯誤。

2. AWS Secrets Manager 為您的 RDS PostgreSQL 資料庫和 Aurora 資料庫設定密碼。然後參考 AWS Identity and Access Management (IAM) 存取政策和角色中的密碼。如需詳細資訊，請參閱 建立秘密和 IAM 角色來使用聯合查詢。

   > **Note**
   >
   > 如果您的叢集使用增強型 VPC 路由，您可能需要為 AWS Secrets Manager設定界面 VPC 端點。當 Amazon Redshift 叢集的 VPC 和子網路無法存取公 AWS Secrets Manager 有端點時，這是必要的。使用 VPC 介面端點時，虛擬私人 VPC 端中 Amazon Redshift 叢集之間的通訊會以私密 AWS Secrets Manager 方式路由到 VPC 點界面。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的建立介面端點。

3. 套用您先前在 Amazon Redshift 叢集中建立的 IAM 角色。如需詳細資訊，請參閱 建立秘密和 IAM 角色來使用聯合查詢。
4. 使用外部結構描述連接到 RDS PostgreSQL 和 Aurora PostgreSQL 資料庫。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。如需如何使用聯合查詢的範例，請參閱使用聯合查詢的範例。

5. 執行參考 RDS PostgreSQL 和 Aurora PostgreSQL 資料庫之外部結構描述的 SQL 查詢。

開始使用與 PostgreSQL 的同盟查詢 AWS CloudFormation

您可以使用聯合查詢在操作資料庫之間進行查詢。在這份入門指南中，您可以使用範例 AWS CloudFormation 堆疊啟用從 Amazon Redshift 叢集到 Aurora PostgreSQL 無伺服器資料庫的聯合查詢來自動設定。您可以快速啟動並執行，而不必執行 SQL 陳述式來佈建資源。

堆疊會建立外部結構描述，並參考您的 Aurora PostgreSQL 執行個體，其中包含具有範例資料的資料表。您可以從 Redshift 叢集中查詢外部結構描述中的資料表。

如果您想要透過執行 SQL 陳述式來設定外部結構描述來開始使用聯合查詢，而不使用 CloudFormation，請參閱開始使用 PostgreSQL 的聯合查詢。

在執行聯合查詢的 CloudFormation 堆疊之前，請確定您已開啟資料 API 的 Amazon Aurora PostgreSQL 相容版無伺服器資料庫。您可以在資料庫屬性中開啟資料 API。如果找不到設定，請再次確認您執行的是 Aurora PostgreSQL 的無伺服器執行個體。此外，請確定您擁有使用 RA3 節點的 Amazon Redshift 叢集。我們建議將 Redshift 叢集和無伺服器 Aurora PostgreSQL 執行個體放在相同的虛擬私有雲端 (VPC) 和子網路群組中。如此一來，您就可以將 Amazon Redshift 叢集的安全群組新增至 Aurora PostgreSQL 資料庫執行個體的安全群組傳入規則。

如需開始設定 Amazon Redshift 叢集的詳細資訊，請參閱 Amazon Redshift 佈建叢集。如需有關使用設定資源的詳細資訊 CloudFormation，請參閱什麼是 AWS CloudFormation？ 。如需有關設定 Aurora 資料庫叢集資料庫的詳細資訊，請參閱建立 Aurora 資料庫叢集無伺服器 v1 資料庫叢集。

啟動用於 Redshift 聯合查詢的 CloudFormation 堆疊

使用下列程序為 Amazon Redshift 啟動您的 CloudFormation 堆疊，以啟用聯合查詢。在執行此操作之前，請確定您已設定 Amazon Redshift 叢集和無伺服器 Aurora PostgreSQL 執行個體。

若要啟動同盟查詢的 CloudFormation 堆疊

1. 按一下此處啟動 CFN 堆疊以啟動中的 CloudFormation服務。 AWS Management Console

   如果出現系統提示，請登入。

   堆疊建立程序會啟動，參考儲存在 Amazon S3 中的 CloudFormation 範本檔案。 CloudFormation 模板是 JSON 格式的文本文件，它聲明構成堆棧的 AWS 資源。
2. 選擇下一步以輸入堆疊詳細資料。

3. 在參數底下，為叢集輸入下列內容：

   - Amazon Redshift 叢集名稱，例如 **ra3-consumer-cluster**
   - 特定的資料庫名稱，例如 **dev**
   - 資料庫使用者的名稱，例如 **consumeruser**

   同時輸入 Aurora DB 叢集資料庫的參數，包括使用者、資料庫名稱、連接埠和端點。我們建議您使用測試叢集和測試無伺服器資料庫，因為堆疊會建立數個資料庫物件。

   選擇下一步。

   堆疊選項隨即出現。
4. 選擇下一步以接受預設設定。
5. 在 [功能] 下，選擇 [我確認 AWS CloudFormation 可能會建立 IAM 資源]。
6. 選擇建立堆疊。

選擇 [建立堆疊]。 CloudFormation 佈建模板資源，這需要大約 10 分鐘，並創建一個外部模式。

如果建立堆疊時發生錯誤，請執行下列動作：

- 檢視 CloudFormation 事件索引標籤，瞭解可協助您解決錯誤的資訊。
- 請確定您輸入正確的 Amazon Redshift 叢集名稱、資料庫名稱和資料庫使用者名稱。同時檢查 Aurora PostgreSQL 執行個體的參數。
- 請確定您的叢集具有 RA3 節點。
- 請確定您的資料庫和 Redshift 叢集位於相同的子網路和安全群組中。

從外部結構描述查詢資料

若要使用下列程序，請確定您在所述叢集和資料庫上有執行查詢所需的許可。

使用聯合查詢來查詢外部資料庫

1. 使用用戶端工具 (例如 Redshift 查詢編輯器) 連線至您在建立堆疊時輸入的 Redshift 資料庫。
2. 查詢堆疊所建立的外部結構描述。

```
select * from svv_external_schemas;
```

SVV_EXTERNAL_SCHEMAS 檢視會傳回可用外部結構描述的相關資訊。在此情況下，會傳回堆疊所建立的外部結構描述 (myfederated_schema)。如果您有任何設定，可能還會傳回其他外部結構描述。檢視也會傳回結構描述的關聯資料庫。資料庫是您在建立堆疊時輸入的 Aurora 資料庫叢集資料庫。堆疊會將資料表新增至 Aurora DB 叢集資料庫 (稱為)category，並將另一個名為的資料表新增sales。

3. 在參考您 Aurora PostgreSQL 資料庫的外部結構描述中的資料表上執行 SQL 查詢。查詢如下列範例所示。

```
SELECT count(*) FROM myfederated_schema.category;
```

category 資料表會傳回幾個記錄。您也可以從 sales 資料表中傳回記錄。

```
SELECT count(*) FROM myfederated_schema.sales;
```

如需更多範例，請參閱使用聯合查詢的範例。

開始使用 MySQL 的聯合查詢

若要建立 MySQL 資料庫的聯合查詢，您可以遵循以下一般方法：

1. 設定從您的 Amazon Redshift 叢集到 Amazon RDS 或 Aurora MySQL DB 執行個體的連線。

   若要這樣做，請確定 RDS MySQL 或 Aurora MySQL DB 執行個體可以接受來自 Amazon Redshift 叢集的連線。我們建議您的 Amazon Redshift 叢集和 Amazon RDS 或 Aurora MySQL 執行個體位於相同的虛擬私有雲端 (VPC) 和子網路群組中。如此一來，您就可以將 Amazon Redshift 叢集的安全群組新增至 RDS 或 Aurora MySQL DB 執行個體的安全群組傳入規則。

   您也可以設定 VPC 對等，或其他允許 Amazon Redshift 與 RDS 或 Aurora MySQL 執行個體建立連線的網路。如需 VPC 網路的相關資訊，請參閱以下內容。

   - 《Amazon VPC 對等互連指南》中的什麼是 VPC 對等互連？
   - 《Amazon RDS 使用者指南》中的在 VPC 中使用 DB 執行個體。

   > **Note**
   >
   > 如果您的 Amazon Redshift 叢集與 RDS 或 Aurora MySQL 執行個體位於不同的 VPC 中，請啟用增強型 VPC 路由。否則，當您執行聯合查詢時，您可能會收到逾時錯誤。

2. AWS Secrets Manager 為您的 RDS MySQL 和 Aurora MySQL 資料庫設定密碼。然後參考 AWS Identity and Access Management (IAM) 存取政策和角色中的密碼。如需詳細資訊，請參閱 建立秘密和 IAM 角色來使用聯合查詢。

   > **Note**
   >
   > 如果您的叢集使用增強型 VPC 路由，您可能需要為 AWS Secrets Manager設定界面 VPC 端點。當 Amazon Redshift 叢集的 VPC 和子網路無法存取公 AWS Secrets Manager 有端點時，這是必要的。當您使用 VPC 界面端點時，VPC 中的 Amazon Redshift 叢集與 AWS Secrets Manager 之間的通訊必須私密地從您的 VPC 路由到端點界面。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的建立介面端點。

3. 套用您先前在 Amazon Redshift 叢集中建立的 IAM 角色。如需詳細資訊，請參閱 建立秘密和 IAM 角色來使用聯合查詢。
4. 使用外部結構描述連接到 RDS MySQL 和 Aurora MySQL 資料庫。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。如需如何使用聯合查詢的範例，請參閱搭配 MySQL 使用聯合查詢的範例。
5. 執行參考 RDS MySQL 和 Aurora MySQL 資料庫之外部結構描述的 SQL 查詢。

建立秘密和 IAM 角色來使用聯合查詢

下列步驟顯示如何建立秘密和 IAM 角色，與聯合查詢搭配使用。

必要條件

確定您具備下列先決條件來建立秘密和 IAM 角色，與聯合查詢搭配使用：

- 具有使用者名稱和密碼驗證的 RDS PostgreSQL、Aurora PostgreSQL DB 執行個體、RDS MySQL 或 Aurora MySQL DB 執行個體。
- 具備支援聯合查詢叢集維護版本的 Amazon Redshift 叢集。

若要使用建立密碼 (使用者名稱和密碼) AWS Secrets Manager

1. 使用擁有 RDS 或 Aurora 資料庫叢集執行個體的帳戶登入 Secrets Manager 主控台。
2. 選擇儲存新機密。
3. 選擇 RDS 資料庫的登入資料圖磚。針對 使用者名稱和密碼，輸入您執行個體的值。確認或為加密金鑰選擇值。然後選擇您秘密將會存取的 RDS 資料庫。

> **Note**
>
> 我們建議您使用預設加密金鑰 (DefaultEncryptionKey)。如果您使用自訂加密金鑰，則必須將用於存取該秘密的 IAM 角色新增為金鑰使用者。

4. 輸入秘密的名稱，使用預設的選擇繼續建立步驟，然後選擇儲存。
5. 檢視您的秘密並記下您建立以識別秘密的秘密 ARN 值。

使用秘密建立安全政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 使用與以下相似的 JSON 來建立政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-
rds-secret-VNenFy"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

若要擷取秘密，您需要列出和讀取動作。我們建議您將資源限制在您建立的特定秘密。若要執行此動作，請使用秘密的 Amazon 資源名稱 (ARN) 來限制資源。您也可以使用 IAM 主控台上的視覺編輯器指定許可和資源。

3. 給予政策名稱並完成建立過程。
4. 導覽至 IAM 角色。
5. 為 Redshift – 可自訂建立 IAM 角色。
6. 將您剛建立的 IAM 政策連接至現有的 IAM 角色，或是建立新的 IAM 角色並連接政策。
7. 在您 IAM 角色的信任關係索引標籤上，確認角色包含信任實體 `redshift.amazonaws.com`。
8. 請注意您建立的角色 ARN。這個 ARN 具備秘密的存取權限。

將 IAM 角色連接至您的 Amazon Redshift 叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集。會列出目前 AWS 區域中您帳戶的叢集。
3. 選擇清單中的叢集名稱以檢視該叢集的詳細資訊。
4. 針對動作，選擇管理 IAM 角色。管理 IAM 角色頁面隨即出現。
5. 將您的 IAM 角色新增到叢集。

使用聯合查詢的範例

下列範例顯示如何執行聯合查詢。使用連接到 Amazon Redshift 資料庫的 SQL 用戶端執行 SQL。

搭配 PostgreSQL 使用聯合查詢的範例

以下範例會示範如何設定聯合查詢，參考 Amazon Redshift 資料庫、Aurora PostgreSQL 資料庫和 Amazon S3。此範例會示範聯合查詢的運作方式。若要在您自己的環境中執行它，請將其變更為符合您的環境。如需執行此動作的先決條件，請參閱開始使用 PostgreSQL 的聯合查詢。

建立參考 Aurora PostgreSQL 資料庫的外部結構描述。

```
CREATE EXTERNAL SCHEMA apg
FROM POSTGRES
DATABASE 'database-1' SCHEMA 'myschema'
```

```
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

建立另一個參考 Amazon S3 的外部結構描述，該結構描述使用 Amazon Redshift Spectrum。同時，將使用結構描述的許可授予 public。

```
CREATE EXTERNAL SCHEMA s3
FROM DATA CATALOG
DATABASE 'default' REGION 'us-west-2'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-S3';

GRANT USAGE ON SCHEMA s3 TO public;
```

顯示 Amazon Redshift 資料表中的資料列數。

```
SELECT count(*) FROM public.lineitem;

  count
----------
25075099
```

顯示 Aurora PostgreSQL 資料表中的資料列數。

```
SELECT count(*) FROM apg.lineitem;

count
-------
11760
```

顯示 Amazon S3 中的資料列數。

```
SELECT count(*) FROM s3.lineitem_1t_part;

   count
------------
6144008876
```

從 Amazon Redshift、Aurora PostgreSQL 和 Amazon S3 建立資料表的檢視。這個檢視是用來執行您的聯合查詢。

```
CREATE VIEW lineitem_all AS
  SELECT
 l_orderkey,l_partkey,l_suppkey,l_linenumber,l_quantity,l_extendedprice,l_discount,l_tax,l_retu
        l_shipdate::date,l_commitdate::date,l_receiptdate::date,
 l_shipinstruct ,l_shipmode,l_comment
  FROM s3.lineitem_1t_part
  UNION ALL SELECT * FROM public.lineitem
  UNION ALL SELECT * FROM apg.lineitem
     with no schema binding;
```

搭配限制結果的述詞，顯示 lineitem_all 檢視中的資料列數。

```
SELECT count(*) from lineitem_all WHERE l_quantity = 10;

   count
-----------
123373836
```

找出每年 1 月某件項目的銷售數。

```
SELECT extract(year from l_shipdate) as year,
       extract(month from l_shipdate) as month,
       count(*) as orders
FROM lineitem_all
WHERE extract(month from l_shipdate) = 1
AND l_quantity < 2
GROUP BY 1,2
ORDER BY 1,2;

 year | month | orders
------+-------+---------
 1992 |     1 |  196019
 1993 |     1 | 1582034
 1994 |     1 | 1583181
 1995 |     1 | 1583919
 1996 |     1 | 1583622
 1997 |     1 | 1586541
 1998 |     1 | 1583198
 2016 |     1 |   15542
 2017 |     1 |   15414
 2018 |     1 |   15527
```

```
 2019 |     1 |     151
```

使用混合大小寫名稱的範例

查詢支援的 PostgreSQL 遠端資料庫，該資料庫具有混合大小寫的資料庫、結構描述、資料表或資料欄名稱，然後將 enable_case_sensitive_identifier 設定為 true。如需此工作階段參數的相關資訊，請參閱 enable_case_sensitive_identifier。

```
SET enable_case_sensitive_identifier TO TRUE;
```

一般而言，資料庫和結構描述名稱是小寫的。下列範例顯示如何連線到支援的 PostgreSQL 遠端資料庫，該資料庫的資料庫和結構描述名稱為小寫，而資料表和資料欄的名稱則混合大小寫。

建立外部結構描述，並使其參考具有小寫資料庫名稱 (dblower) 和小寫結構描述名稱 (schemalower) 的 Aurora PostgreSQL 資料庫。

```
CREATE EXTERNAL SCHEMA apg_lower
FROM POSTGRES
DATABASE 'dblower' SCHEMA 'schemalower'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

在執行查詢的工作階段中，將 enable_case_sensitive_identifier 設定為 true。

```
SET enable_case_sensitive_identifier TO TRUE;
```

執行聯合查詢以從 PostgreSQL 資料庫中選取所有資料。資料表 (MixedCaseTab) 和資料欄 (MixedCaseName) 具有混合大小寫的名稱。結果是一個資料列 (Harry)。

```
select * from apg_lower."MixedCaseTab";
```

```
 MixedCaseName
-------
 Harry
```

下列範例顯示如何連線到支援的 PostgreSQL 遠端資料庫，該資料庫的資料庫、結構描述、資料表和資料欄名稱混合大小寫。

將 enable_case_sensitive_identifier 設定為 true，然後再建立外部結構描述。如果 enable_case_sensitive_identifier 在建立外部結構描述之前未設定為 true，則會發生資料庫不存在錯誤。

建立外部結構描述，並使其參考具有混合大小寫資料庫名稱 (UpperDB) 和結構描述名稱 (UpperSchema) 的 Aurora PostgreSQL 資料庫。

```
CREATE EXTERNAL SCHEMA apg_upper
FROM POSTGRES
DATABASE 'UpperDB' SCHEMA 'UpperSchema'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

執行聯合查詢以從 PostgreSQL 資料庫中選取所有資料。資料表 (MixedCaseTab) 和資料欄 (MixedCaseName) 具有混合大小寫的名稱。結果是一個資料列 (Harry)。

```
select * from apg_upper."MixedCaseTab";
```

```
 MixedCaseName
-------
 Harry
```

搭配 MySQL 使用聯合查詢的範例

以下範例會示範如何設定參考 Aurora MySQL 資料庫的聯合查詢。此範例會示範聯合查詢的運作方式。若要在您自己的環境中執行它，請將其變更為符合您的環境。如需執行此動作的先決條件，請參閱開始使用 MySQL 的聯合查詢。

本範例取決於下列先決條件：

- 在 Aurora MySQL 資料庫的 Secrets Manager 中設定的秘密。IAM 存取政策和角色中會參考此秘密。如需詳細資訊，請參閱 建立秘密和 IAM 角色來使用聯合查詢。
- 已設定連結 Amazon Redshift 和 Aurora MySQL 的安全群組。

建立參考 Aurora MySQL 資料庫的外部結構描述。

```
CREATE EXTERNAL SCHEMA amysql
FROM MYSQL
DATABASE 'functional'
URI 'endpoint to remote hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

執行 Aurora MySQL 資料表的範例 SQL 選取，以顯示 Aurora MySQL 中員工資料表中的一個資料列。

```
SELECT level FROM amysql.employees LIMIT 1;

 level
-------
     8
```

Amazon Redshift 與支援的 PostgreSQL 和 MySQL 資料庫之間的資料類型差異

下表顯示 Amazon Redshift 資料類型對對應 Amazon RDS PostgreSQL 或 Aurora PostgreSQL 資料類型的映射。

Amazon Redshift 資料類型	RDS PostgreSQL 或 Aurora PostgreSQL 資料類型	描述
SMALLINT	SMALLINT	帶正負號的 2 位元組整數
INTEGER	INTEGER	帶正負號的 4 位元組整數
BIGINT	BIGINT	帶正負號的 8 位元組整數
DECIMAL	DECIMAL	可選擇精確度 (有效位數) 的精確數值
REAL	REAL	單精度浮點數

Amazon Redshift 資料類型	RDS PostgreSQL 或 Aurora PostgreSQL 資料類型	描述
DOUBLE PRECISION	DOUBLE PRECISION	雙精度浮點數
BOOLEAN	BOOLEAN	邏輯布林值 (true/false)
CHAR	CHAR	固定長度的字元字串
VARCHAR	VARCHAR	可變長度的字元字串 (使用者定義的限制)
DATE	DATE	日曆日期 (年、月、日)
TIMESTAMP	TIMESTAMP	日期和時間 (未使用時區)
TIMESTAMPTZ	TIMESTAMPTZ	日期和時間 (包含時區)
GEOMETRY	PostGIS GEOMETRY	空間資料

下列 RDS PostgreSQL 和 Aurora PostgreSQL 資料類型會轉換為 Amazon Redshift 中的 VARCHAR(64K)：

- JSON、JSONB
- 陣列
- BIT、BIT VARYING
- BYTEA
- 複合類型
- 日期和時間類型 INTERVAL、TIME、TIME WITH TIMEZONE
- 列舉類型
- 貨幣類型
- 網路地址類型
- 數字類型 SERIAL、BIGSERIAL、SMALLSERIAL 和 MONEY
- 物件識別碼類型

- pg_lsn 類型
- 虛擬類型
- 範圍類型
- 文字搜尋類型
- TXID_SNAPSHOT
- UUID
- XML 類型

下表顯示 Amazon Redshift 資料類型至對應 Amazon RDS MySQL 或 Aurora MySQL 資料類型的映射。

Amazon Redshift 資料類型	RDS MySQL 或 Aurora MySQL 資料類型	描述
BOOLEAN	TINYINT(1)	邏輯布林值 (true 或 false)
SMALLINT	TINYINT(UNSIGNED)	帶正負號的 2 位元組整數
SMALLINT	SMALLINT	帶正負號的 2 位元組整數
INTEGER	SMALLINT UNSIGNED	帶正負號的 4 位元組整數
INTEGER	MEDIUMINT (UNSIGNED)	帶正負號的 4 位元組整數
INTEGER	INT	帶正負號的 4 位元組整數
BIGINT	INT UNSIGNED	帶正負號的 8 位元組整數
BIGINT	BIGINT	帶正負號的 8 位元組整數

Amazon Redshift 資料類型	RDS MySQL 或 Aurora MySQL 資料類型	描述
DECIMAL	BIGINT UNSIGNED	可選擇精確度 (有效位數) 的精確數值
DECIMAL	DECIMAL(M,D)	可選擇精確度 (有效位數) 的精確數值
REAL	FLOAT	單精度浮點數
DOUBLE PRECISION	DOUBLE	雙精度浮點數
CHAR	CHAR	固定長度的字元字串
VARCHAR	VARCHAR	可變長度的字元字串 (使用者定義的限制)
DATE	DATE	日曆日期 (年、月、日)
TIME	TIME	時間 (不含時區)
TIMESTAMP	TIMESTAMP	日期和時間 (未使用時區)
TIMESTAMP	DATETIME	時間 (不含時區)
VARCHAR(4)	YEAR	代表年份的可變長度字元

當 TIME資料超出範圍 (00:00:00 – 24:00:00) 時，就會產生錯誤。

下列 RDS MySQL 和 Aurora MySQL 資料類型會轉換為 Amazon Redshift 中的 VARCHAR(64K)：

- BIT
- BINARY
- VARBINARY
- TINYBLOB、BLOB、MEDIUMBLOB、LONGBLOB
- TINYTEXT、TEXT、MEDIUMTEXT、LONGTEXT

- ENUM
- SET
- SPATIAL

使用 Amazon Redshift 存取聯合資料時的注意事項

某些 Amazon Redshift 功能不支援存取聯合資料。您可以在下面找到相關的限制和考量。

以下是將聯合查詢與 Amazon Redshift 搭配使用時的限制及考量：

- 聯合查詢支援讀取外部資料來源。您無法在外部資料來源中寫入或建立資料庫物件。
- 在某些情況下，您可能會在不同的 AWS 區域存取 Amazon RDS 或 Aurora 資料庫叢集資料庫，而不是 Amazon Redshift。在這些情況下，跨 AWS 區域傳輸資料通常會產生網路延遲和帳單費用。我們建議您將 Aurora 全球資料庫與 Amazon Redshift 叢集位於相同 AWS 區域的本機端點搭配使用。Aurora 全域資料庫使用專用的基礎設施來進行跨越任意兩個 AWS 區域的儲存體類型複寫，其延遲一般低於 1 秒。
- 請考慮存取 Amazon RDS 或 Aurora 資料庫叢集的成本。例如，使用此功能存取 Aurora 資料庫叢集時，Aurora 資料庫叢集費用會根據 IOPS 計算。
- 聯合查詢無法啟用從 RDS 或 Aurora 資料庫叢集存取 Amazon Redshift 的功能。
- 聯合查詢僅適用於同時提供 Amazon Redshift 和 Amazon RDS 或 Aurora 資料庫叢集的 AWS 區域。
- 聯合查詢目前不支援 `ALTER SCHEMA`。若要變更結構描述，請使用 `DROP` 再使用 `CREATE EXTERNAL SCHEMA`。
- 聯合查詢不會使用並行擴展。
- 聯合查詢目前不支援透過 PostgreSQL 外部資料包裝程式存取。
- 對 RDS MySQL 或 Aurora MySQL 的聯合查詢支援 READ COMMITTED 層級上的交易隔離。
- 如果未指定，Amazon Redshift 會連線到連接埠 3306 上的 RDS for MySQL 或 Aurora MySQL。在為 MySQL 建立外部結構描述之前，請先確認 MySQL 連接埠號碼。
- 如果未指定，Amazon Redshift 會連線到連接埠 5432 上的 RDS PostgreSQL 或 Aurora PostgreSQL。在為 PostgreSQL 建立外部結構描述之前，請先確認 PostgreSQL 連接埠號碼。
- 當從 MySQL 擷取 TIMESTAMP 和 DATE 資料類型時，零值會視為 NULL。
- 如果使用 Aurora DB 叢集資料庫讀取器端點，可能會發生「無效的快照集」錯誤。這可以透過以下其中一個方法來避免：

- 使用特定的 Aurora 資料庫叢集執行個體端點 (而非使用 Aurora 資料庫叢集端點)。此方法會針對 PostgreSQL 資料庫的結果使用 REPEATABLE READ 交易隔離。
- 使用 Aurora 資料庫叢集讀取器端點，並針對工作階段設pg_federation_repeatable_read定為 false。此方法會針對 PostgreSQL 資料庫的結果使用 READ COMMITTED 交易隔離。
  如需 Aurora 資料庫叢集讀取器端點的詳細資訊，請參閱 Amazon Aurora Aurora 資料庫叢集端點的類型 (英文)。如需 pg_federation_repeatable_read 的資訊，請參閱「pg_federation_repeatable_read」。

以下是針對 PostgreSQL 資料庫使用聯合查詢時交易的考量：

- 如果查詢由聯合資料表組成，則領導者節點會在遠端資料庫上啟動 READ ONLY REPEATABLE READ 交易。在 Amazon Redshift 交易期間會一直保留此交易。
- 領導者節點透過呼叫 pg_export_snapshot 來建立遠端資料庫的快照，並對受影響的表進行讀取鎖定。
- 運算節點會啟動交易，並使用在領導者節點上建立的快照，向遠端資料庫發出查詢。

支援的聯合資料庫版本

Amazon Redshift 外部結構描述可參考外部 RDS PostgreSQL 或 Aurora PostgreSQL 中的資料庫。當其這樣做時，適用下列限制：

- 建立參考 Aurora 資料庫叢集的外部結構描述時，Aurora PostgreSQL 資料庫必須是 9.6 版或更新版本。
- 建立參考 Amazon RDS 的外部結構描述時，Amazon RDS PostgreSQL 資料庫的版本必須是 9.6 或更新版本。

Amazon Redshift 外部結構描述可參考外部 RDS MySQL 或 Aurora MySQL 中的資料庫。當其這樣做時，適用下列限制：

- 建立參考 Aurora 資料庫叢集的外部結構描述時，Aurora MySQL 資料庫必須是 5.6 版或更新版本。
- 建立參考 Amazon RDS 的外部結構描述時，RDS MySQL 資料庫的版本必須是 5.6 或更新版本。

使用 Amazon Redshift Spectrum 查詢外部資料

使用 Amazon Redshift Spectrum，可以有效率地查詢 Amazon S3 裡的檔案並擷取結構化與半結構化的資料，無需將資料載入 Amazon Redshift 資料表。Redshift Spectrum 採用了大量的平行處理，以便非常快速的執行大型資料庫查詢。大部分的處理發生在 Redshift Spectrum 層，大部分資料則保留在 Amazon S3。可以多個叢集同時查詢 Amazon S3 上的相同資料集，無需為每個叢集複製資料副本。

主題

Amazon Redshift Spectrum 概觀

Amazon Redshift Spectrum 位於獨立於您叢集之外的專屬 Amazon Redshift 伺服器上。Amazon Redshift 會推送許多運算密集型任務 (例如述詞篩選和彙整) 到 Redshift Spectrum 層。因此，與其他查詢相比，Redshift Spectrum 查詢使用的叢集處理容量要低得多。Redshift Spectrum 也能聰明的擴展。根據您的查詢需求，Redshift Spectrum 可能會使用數千個執行個體來運用大規模平行處理。

您可以透過定義檔案結構，並將其做為資料表註冊到外部資料目錄中來建立 Redshift Spectrum 資料表。外部資料型錄可以是 AWS Glue Amazon Athena 隨附的資料目錄，或是您自己的 Apache Hive

中繼存放區。您可以使用資料定義語言 (DDL) 命令，或使用連接到外部資料目錄的任何其他工具，在 Amazon Redshift 建立和管理外部資料表。您的任何 Amazon Redshift 叢集都可以立即變更外部資料目錄。

您也可以選擇在一個或多個欄上對外部資料表進行分割。將分割區定義為外部資料表的一部分可以提高效能。這種改進的出現，是因為 Amazon Redshift 查詢最佳化工具會刪除不包含查詢資料的分割區。

在您定義 Redshift Spectrum 資料表後，您可以像任何其他 Amazon Redshift 資料表一樣查詢和聯結資料表。Redshift Spectrum 不支援外部資料表的更新操作。您可以將紅移頻譜表新增到多個 Amazon Redshift 叢集，並從同一區域中的任何叢集查詢 Amazon S3 上的相同 AWS 資料。當您更新 Amazon S3 資料檔案時，資料可立即從任何 Amazon Redshift 叢集進行查詢。

您存取的 AWS Glue 資料目錄可能會加密以提高安全性。如果目 AWS Glue 錄已加密，則需要 AWS Key Management Service (AWS KMS) 金鑰 AWS Glue 才能存取 AWS Glue 目錄。 AWS Glue 並非所有 AWS 區域均提供目錄加密。如需支援的 AWS 區域清單，請參閱AWS Glue 開發人員指南 AWS Glue中的加密和安全存取。如需有關資 AWS Glue 料目錄加密的詳細資訊，請參閱AWS Glue 開發人員指南中的加密資 AWS Glue 料目錄。

> **Note**
>
> 您無法使用與標準 Amazon Redshift 資料表相同的資源來檢視 Redshift Spectrum 資料表的詳細資訊，例如 PG_TABLE_DEF、STV_TBL_PERM、PG_CLASS，或 information_schema。如果您的商業智慧或分析工具無法識別 Redshift Spectrum 外部資料表，請將您的應用程式設定為查詢 SVV_EXTERNAL_TABLES 與 SVV_EXTERNAL_COLUMNS。

Amazon Redshift Spectrum 區域

除非區域特定文件 AWS 區域 中另有說明，否則可在提供 Amazon Redshift 的地方使用紅移頻譜。如需商業區域的 AWS 區域 可用性 Redshift 請參閱. Amazon Web Services 一般參考

Amazon Redshift Spectrum 考量事項

當您使用 Amazon Redshift Spectrum 時，請注意以下考量事項：

- Amazon Redshift 叢集和 Amazon S3 儲存貯體必須位於同一個 AWS 區域。
- Redshift Spectrum 不支援使用已佈建叢集的增強型 VPC 路由。若要存取您的 Amazon S3 資料，您可能需要執行額外的設定步驟。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Redshift Spectrum 和增強型 VPC 路由。

- Redshift Spectrum 支援 Amazon S3 存取點別名。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的為您的存取點使用儲存貯體型別名。不過，Redshift Spectrum 不支援具有 Amazon S3 存取點別名的 VPC。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Redshift Spectrum 和增強型 VPC 路由。
- 您不能在外部資料表上執行更新或刪除操作。若要在指定的結構描述中建立新外部資料表，您可以使用 CREATE EXTERNAL TABLE。如需 CREATE EXTERNAL TABLE 的相關資訊，請參閱 CREATE EXTERNAL TABLE。若要將 SELECT 查詢結果插入外部目錄上的現有外部資料表，您可以使用 INSERT (外部資料表)。如需 INSERT (外部資料表) 的相關資訊，請參閱 INSERT (外部資料表)。
- 除非您使用為 AWS Glue Data Catalog AWS Lake Formation 啟用的功能，否則您無法控制外部資料表的使用者權限。但是，您可以在外部結構描述上授予和撤銷許可。若要取得有關使用 AWS Lake Formation 的更多資訊，請參閱〈〉使用 Redshift 光譜 AWS Lake Formation。
- 要執行 Redshift Spectrum 查詢，資料庫使用者必須具有在資料庫中建立臨時資料表的許可。下列範例可在資料庫 `spectrumdb` 上授予臨時許可至 `spectrumusers` 使用者群組。

```
grant temp on database spectrumdb to group spectrumusers;
```

  如需詳細資訊，請參閱 GRANT。
- 使用 Athena 資料目錄或資 AWS Glue 料目錄做為中繼資料存放區時，請參閱 Amazon Redshift 管理指南中的配額和限制。
- Redshift Spectrum 不支援使用 Kerberos 的 Amazon EMR。

開始使用 Amazon Redshift Spectrum

在本教學課程中，您將了解如何使用 Amazon Redshift Spectrum 直接從 Amazon S3 上的檔案查詢資料。如果您已有叢集和 SQL 用戶端，則可透過最少的設定完成本教學課程。

> **Note**
>
> Redshift Spectrum 查詢會產生額外費用。在本教學中執行範例查詢的成本是名目成本。如需定價的相關資訊，請參閱 Amazon Redshift Spectrum 定價。

必要條件

若要使用 Redshift Spectrum，您需要一個 Amazon Redshift 叢集和一個連接到您叢集的 SQL 用戶端，以便您執行 SQL 命令。叢集與 Amazon S3 中的資料檔案必須在相同的 AWS 區域。

如需如何建立 Amazon Redshift 叢集的相關資訊，請參閱 Amazon Redshift 入門指南中的 Amazon Redshift 佈建的叢集。如需連線到叢集的方式的相關資訊，請參閱亞馬遜 Redshift 入門指南中的連線到 Amazon Redshift 資料倉儲。

在下面的一些範例中，範例資料位於美國東部 (維吉尼亞北部) 區域 (us-east-1)，因此您需要一個同樣位於 us-east-1 中的叢集。或者，您可以使用 Amazon S3 將資料物件從下列儲存貯體和資料夾複製到叢集所在的儲存貯體： AWS 區域

- `s3://redshift-downloads/tickit/spectrum/customers/*`
- `s3://redshift-downloads/tickit/spectrum/sales_partition/*`
- `s3://redshift-downloads/tickit/spectrum/sales/*`
- `s3://redshift-downloads/tickit/spectrum/salesevent/*`

執行類似下列的 Amazon S3 命令，將位於美國東部 (維吉尼亞北部) 的範例資料複製到您的 AWS 區域。執行命令之前，請在儲存貯體中建立儲存貯體和資料夾，以符合 Amazon S3 複製命令。Amazon S3 複製命令的輸出會確認檔案已複製到所需 AWS 區域的 *bucket-name*。

```
aws s3 cp s3://redshift-downloads/tickit/spectrum/ s3://bucket-name/tickit/spectrum/ --
copy-props none --recursive
```

開始使用 Redshift 頻譜 AWS CloudFormation

除了下列步驟之外，您還可以存取 Redshift 頻譜 DataLake AWS CloudFormation 範本以建立包含可查詢的 Amazon S3 儲存貯體的堆疊。如需詳細資訊，請參閱 啟動您的 AWS CloudFormation 堆疊，然後在 Amazon S3 中查詢您的資料。

逐步開始使用 Redshift Spectrum

若要開始使用 Amazon Redshift Spectrum，請遵循這些步驟：

- 步驟 1. 為 Amazon Redshift 建立 IAM 角色
- 步驟 2：建立 IAM 角色與叢集的關聯

- 步驟 3：建立外部結構描述與外部資料表
- 步驟 4：在 Amazon S3 中查詢您的資料

步驟 1. 為 Amazon Redshift 建立 IAM 角色

您的叢集需要授權才能存取 Amazon Athena 中 AWS Glue 的外部資料目錄，以及 Amazon S3 中的資料檔案。若要提供該授權，請參考連接至叢集的 AWS Identity and Access Management (IAM) 角色。如需將角色與 Amazon Redshift 搭配使用的相關資訊，請參閱使用 IAM 角色授權 COPY 與 UNLOAD 操作。

> **Note**
>
> 在某些情況下，您可以將 Athena 資料目錄移轉至 AWS Glue 資料目錄。如果您的叢集位於受支援的 AWS 區域，且您的 AWS Glue Athena 資料目錄中有 Redshift 頻譜外部表格，則可以執行此操作。若要將資 AWS Glue 料目錄與 Redshift 頻譜搭配使用，您可能需要變更 IAM 政策。如需詳細資訊，請參閱《Athena 使用者指南》中的升級至 AWS Glue 資料目錄。

當您為 Amazon Redshift 建立角色時，請選擇下列其中一個方法：

- 如果您將 Redshift 頻譜與 Athena 資料目錄或 AWS Glue 資料目錄搭配使用，請按照中為 Amazon Redshift 建立 IAM 角色所述的步驟進行。
- 如果您在啟用的情況下使用 Redshift 頻譜 AWS Lake Formation，請按照以下程序中概述的步驟進行操作： AWS Glue Data Catalog
  - 若要使用 AWS Glue Data Catalog 已啟用的功能建立 Amazon Redshift 的 IAM 角色 AWS Lake Formation
  - 授予資料表的 SELECT 許可，以在 Lake Formation 資料庫中進行查詢

為 Amazon Redshift 建立 IAM 角色

1. 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Roles (角色)。
3. 選擇 Create Role (建立角色)。
4. 選擇AWS 服務做為信任的實體，然後選擇Redshift做為使用案例。
5. 在 [其他使用案例] 下 AWS 服務，選擇 [Redshift]-[自訂]，然後選擇 [下一步]。

6. 連接許可政策頁面隨即出現。如果您使用的是 AWS Glue 資料目錄AWSGlueConsoleFullAccess，請選擇AmazonS3ReadOnlyAccess和。或者，如果您使用的是 Athena 資料目錄，則選擇 AmazonAthenaFullAccess。選擇下一步。

   > **Note**
   >
   > AmazonS3ReadOnlyAccess 政策會授予您的叢集對所有 Amazon S3 儲存貯體的唯讀存取權限。若只要授與 AWS 範例資料儲存貯體的存取權，請建立新原則並新增下列權限。
   >
   > ```
   > {
   >     "Version": "2012-10-17",
   >     "Statement": [
   >         {
   >             "Effect": "Allow",
   >             "Action": [
   >                 "s3:Get*",
   >                 "s3:List*"
   >             ],
   >             "Resource": "arn:aws:s3:::redshift-downloads/*"
   >         }
   >     ]
   > }
   > ```

7. 在 Role name (角色名稱) 中輸入角色名稱，例如 **myspectrum_role**。
8. 檢閱資訊，然後選擇 Create role (建立角色)。
9. 在導覽窗格中，選擇角色。選擇新角色的名稱以檢視摘要，然後複製 Role ARN (角色 ARN) 至您的剪貼簿。此值是您剛剛建立的角色 Amazon Resource Name (ARN)。您可以在建立外部資料表時使用該值，以便在 Amazon S3 上參考資料檔案。

若要使用 AWS Glue Data Catalog 已啟用的功能建立 Amazon Redshift 的 IAM 角色 AWS Lake Formation

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇政策。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 選擇 Create policy (建立政策)。

4. 在 JSON 索引標籤上選擇建立政策。
5. 貼上下列 JSON 政策文件，這能授予資料目錄的存取權限，但會拒絕 Lake Formation 的管理員許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RedshiftPolicyForLF",
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "lakeformation:GetDataAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

6. 完成時，選擇 Review (檢閱) 以檢閱該政策。政策驗證程式會回報任何語法錯誤。
7. 在 Review policy (檢閱政策) 頁面的 Name (名稱) 中，輸入 **myspectrum_policy** 來為您所建立的政策命名。輸入 Description (說明) (選用)。檢閱政策 Summary (摘要) 來查看您的政策所授予的許可。然後選擇 Create policy (建立政策) 來儲存您的工作。

   建立政策之後，您可以提供存取權給使用者。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

  建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。
- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。
- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。

- (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循《IAM 使用者指南》的新增許可到使用者 (主控台) 中的指示。

授予資料表的 SELECT 許可，以在 Lake Formation 資料庫中進行查詢

1. 開啟 Lake Formation 主控台，網址為 https://console.aws.amazon.com/lakeformation/。
2. 在導覽窗格中，選擇資料湖許可，然後選擇授予。
3. 遵循AWS Lake Formation 開發人員指南的使用具名資源方法授予資料表權限中的指示。請提供下列資訊：

   - 在IAM 角色中，選擇您所建立的 IAM 角色 `myspectrum_role`。當您執行 Amazon Redshift 查詢編輯器時，其會使用此 IAM 角色來取得資料的許可。

   > **Note**
   >
   > 若要授予啟用 Lake Formation 之資料目錄中的資料表進行查詢的 SELECT 許可，請執行下列操作：
   >
   > - 在 Lake Formation 中註冊資料的路徑。
   > - 在 Lake Formation 中授予使用者該路徑的許可。
   > - 建立的資料表會位於 Lake Formation 中註冊的路徑。

4. 選擇 Grant (授予)。

> **Important**
>
> 根據最佳實務，請僅允許透過 Lake Formation 許可存取基礎 Amazon S3 物件。若要防止未經授權的存取，則可移除授予 Lake Formation 外部 Amazon S3 物件的任何許可。如果您先前在設定 Lake Formation 之前就已存取 Amazon S3 物件，則請移除先前設定的所有 IAM 政策或儲存貯體許可。如需詳細資訊，請參閱將資AWS Glue 料權限升級至 AWS Lake Formation 模型和 Lake Formation 權限。

步驟 2：建立 IAM 角色與叢集的關聯

現在您已擁有授權 Amazon Redshift 為您存取外部資料目錄和 Amazon S3 的 IAM 角色。此時，您必須將該角色與您的 Amazon Redshift 叢集建立關聯。

將 IAM 角色與叢集建立關聯

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您要更新的叢集名稱。
3. 針對動作，選擇管理 IAM 角色。IAM roles (IAM 角色) 頁面隨即出現。
4. 選擇輸入 ARN，然後輸入 ARN 或 IAM 角色，或是從清單中選擇 IAM 角色。然後選擇 Add IAM role (新增 IAM 角色) 來將其新增至 Attached IAM roles (已連接的 IAM 角色) 清單。
5. 選擇 Done (完成) 來將 IAM 角色與叢集建立關聯。叢集會進行修改以完成變更。

步驟 3：建立外部結構描述與外部資料表

在外部結構描述中建立外部資料表。外部結構描述參考外部資料目錄中的資料庫，並提供授權您的叢集代表您存取 Amazon S3 的 IAM 角色 ARN。您可以在 Amazon Athena 資料目錄或 Apache Hive 中繼存放區 (例如 Amazon EMR) 中建立外部資料庫。 AWS Glue Data Catalog在此範例中，當您建立外部結構描述 Amazon Redshift 時，在 Amazon Athena 資料目錄中建立外部資料庫。如需詳細資訊，請參閱 建立 Amazon Redshift Spectrum 外部結構描述。

建立外部結構描述與外部資料表

1. 如要建立外部結構描述，請將以下命令中的 IAM 角色 ARN 替換為您在步驟 1 中建立的角色 ARN。然後，在您的 SQL 用戶端中執行命令。

```
create external schema myspectrum_schema
from data catalog
database 'myspectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
create external database if not exists;
```

2. 若要建立外部資料表，請執行以下 CREATE EXTERNAL TABLE 命令。

> **Note**
>
> 您的叢集和 Amazon S3 儲存貯體必須位於相同的 AWS 區域。在此範例中建立外部表命令，包含範例資料的 Amazon S3 儲存貯體位於美國東部 (維吉尼亞北部) AWS 區域。若要查看來源資料，請下載 sales_ts.000 檔案。。

您可以修改此範例，以便在其他範例中執行 AWS 區域。在您想要的位置建立 Amazon S3 儲存貯體 AWS 區域。使用 Amazon S3 複製命令複製銷售資料。然後將範例 CREATE EXTERNAL TABLE 命令中的位置選項更新到您的儲存貯體。

```
aws s3 cp s3://redshift-downloads/tickit/spectrum/sales/ s3://bucket-name/
tickit/spectrum/sales/ --copy-props none --recursive
```

Amazon S3 複製命令的輸出會確認檔案已複製到所需 AWS 區域的 *bucket-name*。

```
copy: s3://redshift-downloads/tickit/spectrum/sales/sales_ts.000 to
 s3://bucket-name/tickit/spectrum/sales/sales_ts.000
```

```
create external table myspectrum_schema.sales(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales/'
table properties ('numRows'='172000');
```

步驟 4：在 Amazon S3 中查詢您的資料

建立外部資料表後，您可以透過用於查詢其他 Amazon Redshift 資料表的相同 SELECT 陳述式來查詢它們。這些 SELECT 陳述式查詢包括聯結資料表、彙總資料和述詞篩選。

在 Amazon S3 中查詢您的資料

1. 取得 MYSPECTRUM_SCHEMA.SALES 資料表中的列數。

```
select count(*) from myspectrum_schema.sales;
```

```
count
------
172462
```

2. 為符合最佳實務，請將較大的事實資料表存放於 Amazon S3，並將較小的維度資料表存放於 Amazon Redshift。如果您在 [載入資料] 中載入範例資料，您的資料庫中會有一個名為 EVENT 的資料表。如果沒有，請使用以下命令來建立 EVENT 資料表。

```
create table event(
eventid integer not null distkey,
venueid smallint not null,
catid smallint not null,
dateid smallint not null sortkey,
eventname varchar(200),
starttime timestamp);
```

3. 透過將以下 COPY 命令中的 IAM 角色 ARN 替換為您在 步驟 1. 為 Amazon Redshift 建立 IAM 角色 中建立的角色 ARN 來載入 EVENT 資料表。您可以選擇`allevents_pipe.txt`從中的 Amazon S3 儲存貯體下載和檢視來源資料 AWS 區域 `us-east-1`。

```
copy event from 's3://redshift-downloads/tickit/allevents_pipe.txt'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
delimiter '|' timeformat 'YYYY-MM-DD HH:MI:SS' region 'us-east-1';
```

以下範例會結合使用外部 Amazon S3 資料表 MYSPECTRUM_SCHEMA.SALES 與本機 Amazon Redshift 資料表 EVENT，藉此尋找排名前十名事件的總銷售額。

```
select top 10 myspectrum_schema.sales.eventid,
 sum(myspectrum_schema.sales.pricepaid) from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
eventid | sum
--------+---------
    289 | 51846.00
```

```
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
   6471 | 47997.00
   2118 | 47863.00
    984 | 46780.00
   7851 | 46661.00
   5638 | 46280.00
```

4. 檢視先前查詢的查詢計畫。請注意針對 Amazon S3 上的資料執行的 S3 Seq Scan、S3 HashAggregate 和 S3 Query Scan 步驟。

```
explain
select top 10 myspectrum_schema.sales.eventid,
 sum(myspectrum_schema.sales.pricepaid)
from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
QUERY PLAN


------------------------------------------------------------------------------
XN Limit  (cost=1001055770628.63..1001055770628.65 rows=10 width=31)


  ->  XN Merge  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


        Merge Key: sum(sales.derived_col2)


        ->  XN Network  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


              Send to leader
```

```
               ->  XN Sort  (cost=1001055770628.63..1001055770629.13 rows=200
width=31)

                     Sort Key: sum(sales.derived_col2)


                     ->  XN HashAggregate  (cost=1055770620.49..1055770620.99
rows=200 width=31)

                           ->  XN Hash Join DS_BCAST_INNER
(cost=3119.97..1055769620.49 rows=200000 width=31)

                                 Hash Cond: ("outer".derived_col1 = "inner".eventid)


                                 ->  XN S3 Query Scan sales  (cost=3010.00..5010.50
rows=200000 width=31)

                                       ->  S3 HashAggregate  (cost=3010.00..3010.50
rows=200000 width=16)

                                             ->  S3 Seq Scan myspectrum_schema.sales
location:"s3://redshift-downloads/tickit/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)
                                                   Filter: (pricepaid > 30.00)


                                 ->  XN Hash  (cost=87.98..87.98 rows=8798 width=4)


                                       ->  XN Seq Scan on event  (cost=0.00..87.98
rows=8798 width=4)
```

啟動您的 AWS CloudFormation 堆疊，然後在 Amazon S3 中查詢您的資料

建立 Amazon Redshift 叢集並連線到叢集之後，您可以安裝 Redshift 頻譜 DataLake AWS CloudFormation 範本，然後查詢您的資料。

CloudFormation 安裝 Redshift 頻譜入門 DataLake 範本，並建立包含下列項目的堆疊：

- 名為 myspectrum_role 且與您的 Redshift 叢集相關聯的角色
- 名為 myspectrum_schema 的外部結構描述

- Amazon S3 儲存貯體中名為 sales 的外部資料表
- 名為 event 且已載入資料的 Redshift 資料表

若要啟動您的 Redshift 頻譜入門堆疊 DataLake CloudFormation

1. 選擇啟動 CFN 堆疊。主 CloudFormation 控台會開啟，並選取 DataLake .yml 範本。

   您也可以下載並自訂 Redshift 頻譜入門 DataLake CloudFormation CFN 範本，然後開啟 CloudFormation 主控台 (https://console.aws.amazon.com/cloudformation) 並使用自訂範本建立堆疊。
2. 選擇下一步。
3. 在參數下，輸入 Amazon Redshift 叢集名稱、資料庫名稱和您的資料庫使用者名稱。
4. 選擇下一步。

   堆疊選項隨即出現。
5. 選擇下一步以接受預設設定。
6. 檢閱資訊並在 [功能] 下方，選擇 [我確認 AWS CloudFormation 可能會建立 IAM 資源]。
7. 選擇建立堆疊。

如果在建立堆疊時發生錯誤，請參閱下列資訊：

- 檢視 CloudFormation 事件索引標籤，瞭解可協助您解決錯誤的資訊。
- 請先刪除 DataLake CloudFormation 堆疊，然後再次嘗試此作業。
- 請確定您已連線到 Amazon Redshift 資料庫。
- 請確定您為 Amazon Redshift 叢集名稱、資料庫名稱和資料庫使用者名稱輸入正確的資訊。

在 Amazon S3 中查詢您的資料

您可以使用用於查詢其他 Amazon Redshift 資料表的相同 SELECT 陳述式，來查詢外部資料表。這些 SELECT 陳述式查詢包括聯結資料表、彙總資料和述詞篩選。

下列查詢會傳回 myspectrum_schema.sales 外部資料表中的列數。

```
select count(*) from myspectrum_schema.sales;
```

```
count
```

```
------
172462
```

聯結外部資料表與本機資料表

以下範例會結合使用外部資料表 myspectrum_schema.sales 與本機資料表 event，以尋找前十名事件的總銷售額。

```
select top 10 myspectrum_schema.sales.eventid, sum(myspectrum_schema.sales.pricepaid)
 from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
eventid | sum
--------+---------
    289 | 51846.00
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
   6471 | 47997.00
   2118 | 47863.00
    984 | 46780.00
   7851 | 46661.00
   5638 | 46280.00
```

檢視查詢計劃

檢視先前查詢的查詢計畫。請注意在 Amazon S3 的資料上執行的 S3 Seq Scan、S3 HashAggregate 和 S3 Query Scan 步驟。

```
explain
select top 10 myspectrum_schema.sales.eventid, sum(myspectrum_schema.sales.pricepaid)
from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
QUERY PLAN

------------------------------------------------------------------------------
XN Limit  (cost=1001055770628.63..1001055770628.65 rows=10 width=31)

  ->  XN Merge  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

        Merge Key: sum(sales.derived_col2)

        ->  XN Network  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

              Send to leader

              ->  XN Sort  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

                    Sort Key: sum(sales.derived_col2)

                    ->  XN HashAggregate  (cost=1055770620.49..1055770620.99 rows=200
 width=31)

                          ->  XN Hash Join DS_BCAST_INNER  (cost=3119.97..1055769620.49
 rows=200000 width=31)

                                Hash Cond: ("outer".derived_col1 = "inner".eventid)

                                ->  XN S3 Query Scan sales  (cost=3010.00..5010.50
 rows=200000 width=31)

                                      ->  S3 HashAggregate  (cost=3010.00..3010.50
 rows=200000 width=16)
```

```
                                          ->  S3 Seq Scan spectrum.sales
location:"s3://redshift-downloads/tickit/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)
                                                Filter: (pricepaid > 30.00)


                               ->  XN Hash  (cost=87.98..87.98 rows=8798 width=4)


                                     ->  XN Seq Scan on event  (cost=0.00..87.98
rows=8798 width=4)
```

Amazon Redshift Spectrum 的 IAM 政策

根據預設，Amazon Redshift Spectrum 會在支援 AWS Glue的 AWS 區域 AWS Glue Data Catalog 中使用。在其他 AWS 地區，Redshift 頻譜使用 Athena 資料目錄。您的叢集需要授權才能存取 AWS Glue 或 Athena 中的外部資料目錄，以及 Amazon S3 中的資料檔案。您可以參考附加至叢集的 AWS Identity and Access Management (IAM) 角色來提供該授權。如果您使用 Apache Hive 中繼存放區來管理資料目錄，則無需提供對 Athena 的存取權限。

您可以鏈結角色，以便您的叢集可以擔任未連接到叢集的其他角色。如需詳細資訊，請參閱 在 Amazon Redshift Spectrum 中鏈結 IAM 角色。

您存取的 AWS Glue 目錄可能會加密以提高安全性。如果目 AWS Glue 錄已加密，則需要用於存 AWS Glue 取 AWS Glue 資料目錄的 AWS KMS 金鑰。如需詳細資訊，請參閱AWS Glue 開發人員指南中的加密資 AWS Glue 料目錄。

主題

- Amazon S3 許可
- 跨帳戶 Amazon S3 許可
- 授予或限制使用 Redshift Spectrum 存取的政策
- 授予最低許可的政策
- 在 Amazon Redshift Spectrum 中鏈結 IAM 角色
- 控制對 AWS Glue 資料目錄的存取

Amazon S3 許可

您的叢集至少需要對您 Amazon S3 儲存貯體的 GET 和 LIST 存取權限。如果您的儲存貯體與叢集不在同一個 AWS 帳戶中，您的儲存貯體也必須授權叢集存取資料。如需詳細資訊，請參閱授權 Amazon Redshift 代表您存取其他 AWS 服務。

> **Note**
>
> Amazon S3 儲存貯體不能使用僅從特定 VPC 端點限制存取的儲存貯體政策。

下列政策會授予對任何 Amazon S3 儲存貯體的 GET 和 LIST 存取權限。該政策允許 Redshift Spectrum 對 Amazon S3 儲存貯體的存取權以及 COPY 操作。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": ["s3:Get*", "s3:List*"],
  "Resource": "*"
 }]
}
```

下列政策會授予對您名為 myBucket 之 Amazon S3 儲存貯體的 GET 和 LIST 存取權限。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": ["s3:Get*", "s3:List*"],
  "Resource": "arn:aws:s3:::myBucket/*"
 }]
}
```

跨帳戶 Amazon S3 許可

若要授與 Redshift 頻譜存取屬於其他 AWS 帳戶之 Amazon S3 儲存貯體中資料的權限，請將以下政策新增至 Amazon S3 儲存貯體。如需詳細資訊，請參閱授予跨帳戶儲存貯體許可。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "Example permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::redshift-account:role/spectrumrole"
            },
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::bucketname",
                "arn:aws:s3:::bucketname/*"
            ]
        }
    ]
}
```

授予或限制使用 Redshift Spectrum 存取的政策

若要授予只使用 Redshift Spectrum 的 Amazon S3 儲存貯體存取權，請加入允許使用者代理程式 AWS Redshift/Spectrum 存取的條件。以下政策僅允許 Redshift Spectrum 存取 Amazon S3 儲存貯體。它排除了其他存取權限，例如 COPY 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["s3:Get*", "s3:List*"],
        "Resource": "arn:aws:s3:::myBucket/*",
                "Condition": {"StringEquals": {"aws:UserAgent": "AWS Redshift/
Spectrum"}}
    }]
}
```

同樣的，您可能希望建立一個允許存取 COPY 操作，但不包括 Redshift Spectrum 存取的 IAM 角色。若要這麼做，請加入拒絕使用者代理程式 **AWS Redshift/Spectrum** 存取的條件。以下政策允許存取除 Redshift Spectrum 之外的 Amazon S3 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": ["s3:Get*", "s3:List*"],
        "Resource": "arn:aws:s3:::myBucket/*",
                "Condition": {"StringNotEquals": {"aws:UserAgent": "AWS Redshift/
Spectrum"}}
    }]
}
```

授予最低許可的政策

下列政策授予將 Redshift 頻譜與 Amazon S3 和 Athena 搭配使用所需的最低許可。 AWS Glue

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::bucketname",
                "arn:aws:s3:::bucketname/folder1/folder2/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:DeleteDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:CreateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
```

```
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

如果您使用 Athena 做為資料目錄而非使用 AWS Glue，則此原則需要完整的 Athena 存取權。下列政策會授予對 Athena 資源的存取許可。如果外部資料庫位於 Hive 中繼存放區中，則您不需要 Athena 存取權限。

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": ["athena:*"],
  "Resource": ["*"]
 }]
}
```

在 Amazon Redshift Spectrum 中鏈結 IAM 角色

將角色附加到叢集時，您的叢集可以擔任該角色來存取 Amazon S3、Athena，並 AWS Glue 代表您存取。如果連接至叢集的角色無法存取必要資源，您可鏈結其他角色，該角色甚至可以來自其他帳戶。您的叢集接著會暫時擔任鏈結的角色，以存取資料。您也可以用鏈結角色的方式來授予跨帳戶存取。您最多可以鏈結 10 個角色。鏈結中的每個角色都會擔任鏈結中的下一個角色，直到叢集擔任鏈結尾端的角色為止。

若要鏈結角色，您應建立角色間的信任關係。擔任其他角色的角色，必須擁有允許其擔任指定角色的許可政策。另一方面，要傳送許可的角色必須擁有信任政策，允許其將許可傳送給其他鏈結的角色。如需詳細資訊，請參閱在 Amazon Redshift 中鏈結 IAM 角色。

當您執行 CREATE EXTERNAL SCHEMA 命令時，您可包括逗號分隔的角色 ARN 清單來鏈結角色。

> **Note**
>
> 鏈結的角色清單不得包含空格。

在以下範例中，MyRedshiftRole 連接到叢集。MyRedshiftRole 會擔任角色 AcmeData，它屬於帳戶 111122223333。

```
create external schema acme from data catalog
database 'acmedb' region 'us-west-2'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole,arn:aws:iam::111122223333:role/
AcmeData';
```

控制對 AWS Glue 資料目錄的存取

如果您用 AWS Glue 於資料目錄，則可以透過 IAM 政策對 AWS Glue 資料目錄套用精細的存取控制。例如，您可能只想要對特定 IAM 角色公開一些資料庫和資料表。

以下各節說明存取資料目錄中儲存之資料的各種層級存取 AWS Glue 權管理政策。

主題

- 資料庫操作的政策
- 資料表操作的政策
- 分割區操作的政策

資料庫操作的政策

如果您想要授與使用者檢視和建立資料庫的權限，他們需要資料庫和 AWS Glue 資料目錄的存取權限。

下列範例查詢會建立資料庫。

```
CREATE EXTERNAL SCHEMA example_db
FROM DATA CATALOG DATABASE 'example_db' region 'us-west-2'
IAM_ROLE 'arn:aws:iam::redshift-account:role/spectrumrole'
CREATE EXTERNAL DATABASE IF NOT EXISTS
```

以下 IAM 政策會提供用於建立資料庫所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:catalog"
            ]
        }
    ]
}
```

下列範例查詢會列出目前的資料庫。

```
SELECT * FROM SVV_EXTERNAL_DATABASES WHERE
databasename = 'example_db1' or databasename = 'example_db2';
```

以下 IAM 政策會提供用於列出目前的資料庫所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "glue:GetDatabases"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:database/example_db1",
                "arn:aws:glue:us-west-2:redshift-account:database/example_db2",
                "arn:aws:glue:us-west-2:redshift-account:catalog"

            ]
        }
    ]
}
```

資料表操作的政策

如果要提供使用者檢視、建立、卸除、變更或對資料表採取其他動作的許可，使用者便需要數種類型的存取。使用者需要自行存取資料表、其所屬的資料庫，以及目錄。

以下範例查詢會建立外部資料表。

```
CREATE EXTERNAL TABLE example_db.example_tbl0(
    col0 INT,
    col1 VARCHAR(255)
) PARTITIONED BY (part INT) STORED AS TEXTFILE
LOCATION 's3://test/s3/location/';
```

以下 IAM 政策會提供用於建立外部資料表所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateTable"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:catalog",
```

```
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
            ]
        }
    ]
}
```

以下每個範例查詢會列出目前的外部資料表。

```
SELECT * FROM svv_external_tables
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

```
SELECT * FROM svv_external_columns
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

```
SELECT parameters FROM svv_external_tables
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

以下 IAM 政策會提供用於列出目前的外部資料表所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:catalog",
```

```
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/
example_tbl0",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl1"

            ]
        }
    ]
}
```

以下範例查詢會修改現有資料表。

```
ALTER TABLE example_db.example_tbl0
SET TABLE PROPERTIES ('numRows' = '100');
```

以下 IAM 政策會提供用於修改現有資料表所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTable",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:catalog",
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"

            ]
        }
    ]
}
```

以下範例查詢會捨棄現有資料表。

```
DROP TABLE example_db.example_tbl0;
```

以下 IAM 政策會提供用於捨棄現有資料表所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:DeleteTable"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:catalog",
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
            ]
        }
    ]
}
```

分割區操作的政策

如果要提供使用者執行分割區層級操作 (檢視、建立、捨棄、變更等等) 的許可，使用者需要分割區所屬資料表的許可。他們也需要相關資料庫和 AWS Glue 資料目錄的許可。

下列範例查詢會建立分割區。

```
ALTER TABLE example_db.example_tbl0
ADD PARTITION (part=0) LOCATION 's3://test/s3/location/part=0/';
ALTER TABLE example_db.example_t
ADD PARTITION (part=1) LOCATION 's3://test/s3/location/part=1/';
```

以下 IAM 政策會提供用於建立分割區所需的最低許可。

```
 {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetTable",
                "glue:BatchCreatePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:catalog",
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
            ]
        }
    ]
}
```

下列範例查詢會列出目前的分割區。

```
SELECT * FROM svv_external_partitions
WHERE schemname = 'example_db' AND
tablename = 'example_tbl0'
```

以下 IAM 政策會提供用於列出目前的分割區所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetPartitions",
                "glue:GetTables",
                "glue:GetTable"
            ],
            "Resource": [
```

```
                "arn:aws:glue:us-west-2:redshift-account:catalog",
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
            ]
        }
    ]
}
```

以下範例查詢會修改現有分割區。

```
ALTER TABLE example_db.example_tbl0 PARTITION(part='0')
SET LOCATION 's3://test/s3/new/location/part=0/';
```

以下 IAM 政策會提供用於修改現有分割區所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetPartition",
                "glue:UpdatePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:catalog",
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
            ]
        }
    ]
}
```

以下範例查詢會捨棄現有分割區。

```
ALTER TABLE example_db.example_tbl0 DROP PARTITION(part='0');
```

以下 IAM 政策會提供刪除現有分割區所需的最低許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:DeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:redshift-account:catalog",
                "arn:aws:glue:us-west-2:redshift-account:database/example_db",
                "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
            ]
        }
    ]
}
```

使用 Redshift 光譜 AWS Lake Formation

您可以使用針 AWS Lake Formation 對存放在 Amazon S3 中的資料集中定義和強制執行資料庫、表格和欄層級存取政策。透過啟用 Lake Formation 的 AWS Glue Data Catalog 註冊資料後，您即可使用 Redshift Spectrum 等數種服務來查詢該資料。

Lake Formation 提供資料目錄的安全功能和管理能力。在 Lake Formation 中，您能夠授予和撤銷資料庫、資料表、欄和基礎 Amazon S3 儲存體等資料目錄物件的許可。

> **Important**
>
> 在可以使用湖泊形成的 AWS 區域中，您只能在啟用 Lake Formation 的資料目錄下使用 Redshift 光譜。如需可用區域的清單，請參閱AWS 一般參考中的AWS Lake Formation 端點和配額。

透過使用 Redshift Spectrum 搭配 Lake Formation，您可以執行以下操作：

- 使用 Lake Formation 做為集中位置，您可以在其中授予和撤銷對資料湖中所有資料的許可和存取控制許可。Lake Formation 提供許可階層，用以控制對資料目錄中資料庫和資料表的存取權。如需詳細資訊，請參閱《AWS Lake Formation 開發人員指南》中的 Lake Formation 許可概觀。
- 建立外部資料表並在資料湖中的資料上執行查詢 資料湖帳戶管理員必須先向 Lake Formation 註冊包含來源資料的現有 Amazon S3 路徑，帳戶中的使用者才可以執行查詢。管理員還需建立資料表，並授予使用者相關許可。您可授予資料庫、資料表或資料欄層級的存取權限。管理員可以使用 Lake Formation 中的資料篩選條件，對儲存在 Amazon S3 中的敏感資料授予精細的存取控制。如需詳細資訊，請參閱 使用資料篩選條件來實現列層級和儲存格層級安全。

  在資料目錄中註冊資料之後，每當使用者嘗試執行查詢時，Lake Formation 都會驗證該特定主體對資料表的存取權。Lake Formation 會將暫臨時憑證提供給 Redshift Spectrum，並執行查詢。
- 使用使用或取得的 IAM 登入資料對自動掛載執行 Redshift 頻譜查詢`GetClusterCredentials`，並依資料庫 AWS Glue Data Catalog 使用者 (IAMR: 使用者名稱`GetCredentials`或 IAM: 使用者名稱) 管理 Lake Formation 許可。

當您將 Redshift Spectrum 搭配為 Lake Formation 啟用的資料目錄一起使用時，必須具備以下其中一項：

- 與具有資料目錄許可的叢集關聯的 IAM 角色。
- 設定用來管理外部資源存取權的聯合 IAM 身分。如需詳細資訊，請參閱使用聯合身分來管理 Amazon Redshift 對本機資源的存取和 Amazon Redshift 外部資料表的存取。

> **Important**
>
> 將 Redshift Spectrum 搭配為 Lake Formation 啟用的資料目錄一起使用時，您無法鏈結 IAM 角色。

若要進一步瞭解設定與 Redshift 頻譜搭配使用所需的步驟，請參閱AWS Lake Formation 開發人員指南中的〈Lake Formation〉中的〈教學課程：從 JDBC 來源建立資料湖〉。 AWS Lake Formation 具體而言，請參閱使用 Amazon Redshift Spectrum 查詢資料湖中的資料，以了解與 Redshift Spectrum 整合的詳細資訊。本主題中使用的資料和 AWS 資源取決於自學課程中先前的步驟。

使用資料篩選條件來實現列層級和儲存格層級安全

您可以在中定義資料篩選器， AWS Lake Formation 以控制 Redshift Spectrum 查詢對於資料目錄中定義之資料的列層級和儲存格層級存取。若要進行此設定，您需要執行以下任務：

- 使用以下資訊在 Lake Formation 中建立資料篩選條件：
  - 欄規格，內含要包含在查詢結果中或從查詢結果中排除的欄清單。
  - 列篩選運算式，指定要包含在查詢結果中的列。

  如需如何建立資料篩選條件的相關資訊，請參閱《AWS Lake Formation 開發人員指南》中的 Lake Formation 中的資料篩選條件。
- 在 Amazon Redshift 中建立外部資料表，該資料表會參考已啟用 Lake Formation 的資料目錄中的資料表。如需如何使用 Redshift Spectrum 查詢 Lake Formation 資料表的詳細資訊，請參閱《AWS Lake Formation 開發人員指南》中的使用 Amazon Redshift Spectrum 查詢資料湖中的資料。

在 Amazon Redshift 中定義資料表之後，您可以查詢 Lake Formation 資料表，以及存取僅限資料篩選條件允許的列和欄。

如需如何在 Lake Formation 中設定列層級和儲存格層級安全，然後使用 Redshift Spectrum 進行查詢的詳細指南，請參閱使用 Amazon Redshift Spectrum 搭配 AWS Lake Formation中定義的列層級和儲存格層級安全政策。

在 Amazon Redshift Spectrum 為查詢建立資料檔案

您在 Amazon Redshift Spectrum 中用於查詢的資料檔案，通常與您用於其他應用程式的檔案為相同類型。例如，與 Amazon 雅典娜，Amazon EMR 和亞馬遜一起使用相同類型的文件。 QuickSight 您可以直接從 Amazon S3 查詢原始格式的資料。若要執行此作業，資料檔案的格式必須是 Redshift Spectrum 支援的格式，且位於您叢集可存取的 Amazon S3 儲存貯體中。

包含資料檔案和 Amazon Redshift 叢集的 Amazon S3 儲存貯體必須位於同一個 AWS 區域。如需支援 AWS 區域的相關資訊，請參閱Amazon Redshift Spectrum 區域。

Redshift Spectrum 的資料格式

Redshift Spectrum 支援下列結構化與半結構化的資料格式。

檔案格式	單欄式	支援平行讀取	分割單位
Parquet	是	是	資料列群組
ORC	是	是	Stripe
RCFile	是	是	資料列群組
TextFile	否	是	Row
SequenceFile	否	是	資料列或區塊
RegexSerde	否	是	Row
OpenCSV	否	是	Row
AVRO	否	是	區塊
Ion	否	否	N/A
JSON	否	否	N/A

在上表中，標題指出下列項目：

- 單欄式 – 檔案格式是否實體上以資料欄導向的結構儲存資料，而不是以資料列導向的結構。
- 支援平行讀取 – 檔案格式是否支援讀取檔案內的個別區塊。讀取個別區塊可以跨多個獨立 Redshift Spectrum 請求分配檔案的處理，而不必在單一請求中讀取完整檔案。
- 分割單位 – 對於可平行讀取的檔案格式，分割單位是單一 Redshift Spectrum 請求可處理的最小資料區塊。

> **Note**
>
> 文字檔案中的時間戳記值格式必須為 `yyyy-MM-dd HH:mm:ss.SSSSSS`，如以下的時間戳記值所示：`2017-05-01 11:30:59.000000`。

我們建議使用例如 Apache Parquet 的單欄式儲存檔案格式。使用單欄式儲存檔案格式，您只需選擇所需的資料欄，即可最大幅度地減少 Amazon S3 中的資料傳輸。

Redshift Spectrum 的壓縮類型

為減少儲存空間，提高效能並降低成本，我們強烈建議您壓縮您的資料檔案。Redshift Spectrum 會根據副檔名識別檔案壓縮類型。

Redshift Spectrum 支援下列壓縮類型與副檔名。

壓縮演算法	副檔名	支援平行讀取
Gzip	.gz	否
Bzip2	.bz2	是
Snappy	.snappy	否

您可以套用不同層級的壓縮。最常見的是，您可以壓縮整個檔案或壓縮檔案中的個別區塊。在檔案層級壓縮單欄式格式不會產生效能優勢。

為了使 Redshift Spectrum 能夠平行讀取檔案，下列條件必須成立：

- 檔案格式支援平行讀取。
- 檔案層級壓縮 (如果有的話) 支援平行讀取。

檔案內的個別分割單位是否使用可平行讀取的壓縮演算法進行壓縮並不重要，因為每個分割單位都是由單一 Redshift Spectrum 請求處理。此情況的範例是 Snappy 壓縮的 Parquet 檔案。Parquet 檔案中的個別資料列群組會使用 Snappy 進行壓縮，但檔案的頂層結構仍會保持未壓縮狀態。在此情況下，可以平行讀取檔案，因為每個 Redshift Spectrum 請求都可以從 Amazon S3 中讀取和處理個別列群組。

Redshift Spectrum 的加密

Redshift Spectrum 可透明的解密使用以下加密選項加密的資料檔案：

- 使用由 Amazon S3 管理的 AES-256 加密金鑰的伺服器端加密 (SSE-S3)。
- 使用由 AWS Key Management Service (SSE-KMS) 管理的金鑰進行伺服器端加密。

Redshift Spectrum 不支援 Amazon S3 用戶端加密。如需伺服器端加密的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的使用伺服器端加密保護資料。

Amazon Redshift 使用大量平行處理 (MPP) 來達成在大量資料上操作複雜查詢的快速執行。Redshift Spectrum 擴展了相同的原則來查詢外部資料，並依需要使用多個 Redshift Spectrum 執行個體來掃描檔案。將檔案放在每個資料表的單獨資料夾中。

您可以透過執行下列動作來最佳化資料以進行平行處理：

- 如果您的檔案格式或壓縮不支援平行讀取，請將大型檔案分成許多較小的檔案。建議您使用介於 64 MB 與 1 GB 之間的檔案大小。
- 將所有檔案維持相同大小。如果某些檔案比其他檔案大得多，則 Redshift Spectrum 無法平均分配工作負載。

建立 Amazon Redshift Spectrum 外部結構描述

所有外部資料表必須建立在使用 CREATE EXTERNAL SCHEMA 陳述式建立的外部結構描述中。

> **Note**
>
> 某些應用程式會交互使用 database (資料庫) 和 schema (結構描述) 詞彙。在 Amazon Redshift，我們使用術語「結構描述」。

Amazon Redshift 外部結構描述參考了外部資料目錄中的外部資料庫。您可以在 Amazon Redshift、Amazon Athena、AWS Glue Data Catalog 或 Apache Hive 中繼存放區 (例如 Amazon EMR) 中建立外部資料庫。若您在 Amazon Redshift 中建立了外部資料庫，該資料庫會位於 Athena 資料目錄中。若要在 Hive 中繼存放區中建立資料庫，您需要在 Hive 應用程式中建立資料庫。

Amazon Redshift 需要授權才能代表您存取 Athena 中的資料目錄以及 Amazon S3 中的資料檔案。若要提供該授權，您必須先建立 AWS Identity and Access Management (IAM) 角色。然後將該角色連接至您的叢集，並在 Amazon Redshift `CREATE EXTERNAL SCHEMA` 陳述式中為角色提供 Amazon Resource Name (ARN)。如需授權的相關資訊，請參閱 Amazon Redshift Spectrum 的 IAM 政策。

> **Note**
>
> 如果您目前在 Athena 資料目錄中有 Redshift 頻譜外部表格，則可以將您的 Athena 資料目錄移轉至 AWS Glue 資料目錄。若要將 AWS Glue 資料目錄與 Redshift 頻譜搭配使用，您可能需要變更 IAM 政策。如需詳細資訊，請參閱 Amazon Athena 使用者指南中的升級至資 AWS Glue 料目錄。

若要在建立外部結構描述的同時建立外部資料庫，請指定 FROM DATA CATALOG 並在您的 CREATE EXTERNAL DATABASE 陳述式中包含 CREATE EXTERNAL SCHEMA 子句。

以下範例使用外部資料庫 spectrum_schema 建立名為 spectrum_db 的外部結構描述。

```
create external schema spectrum_schema from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
create external database if not exists;
```

如果使用 Athena 管理資料目錄，請指定 Athena 資料庫名稱和 Athena 資料目錄所在的 AWS 區域。

以下範例使用 Athena 資料目錄中的預設 sampledb 資料庫建立外部結構描述。

```
create external schema athena_schema from data catalog
database 'sampledb'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
region 'us-east-2';
```

> **Note**
>
> 此region參數會參考 Athena 資料目錄所在的 AWS 區域，而不是 Amazon S3 中資料檔案的位置。

如果使用 Hive 中繼存放區 (例如 Amazon EMR) 管理資料目錄，則必須將安全群組設定為允許叢集之間的流量。

在 CREATE EXTERNAL SCHEMA 陳述式中，指定 FROM HIVE METASTORE 並包括該中繼存放區的 URI 與連接埠號碼。以下範例使用名為 hive_db 的 Hive 中繼存放區資料庫建立外部結構描述。

```
create external schema hive_schema
from hive metastore
database 'hive_db'
uri '172.10.10.10' port 99
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
```

若要檢視您叢集的外部結構描述，請查詢 PG_EXTERNAL_SCHEMA 目錄資料表或 SVV_EXTERNAL_SCHEMAS 畫面。下列範例查詢了加入 PG_EXTERNAL_SCHEMA 和 PG_NAMESPACE 的 SVV_EXTERNAL_SCHEMAS。

```
select * from svv_external_schemas
```

如需完整的命令語法與範例，請參閱 CREATE EXTERNAL SCHEMA。

在 Amazon Redshift Spectrum 中使用外部目錄

Amazon Redshift Spectrum 外部資料庫和外部資料表的中繼資料儲存於外部資料目錄中。Redshift Spectrum 中繼資料預設儲存於 Athena 資料目錄中。您可在 Athena 主控台中檢視並管理 Redshift Spectrum 資料庫和資料表。

您也可以使用 Hive 資料定義語言 (DDL)、使用 Athena 或 Hive 中繼存放區 (如 Amazon EMR) 建立並管理外部資料庫和外部資料表。

> **Note**
>
> 我們建議使用 Amazon Redshift 來建立並管理 Redshift Spectrum 中的外部資料庫和外部資料表。

檢視 Athena 和 Redshift 頻譜資料庫 AWS Glue

您可以透過在您的 CREATE EXTERNAL SCHEMA 陳述式中包含 CREATE EXTERNAL DATABASE IF NOT EXISTS，來建立外部資料庫。在這類案例中，外部資料庫的中繼資料會儲存在您的資料目錄內。您透過外部結構描述所建立的外部資料表中繼資料，也會儲存在資料目錄中。

Athena 並為每個支 AWS Glue 持維護數據目錄 AWS 區域。若要檢視表格中繼資料，請登入 Athena 或 AWS Glue 主控台。在 Athena 中，選擇資料來源，您的 AWS Glue，然後檢視資料庫的詳細資料。在中 AWS Glue，選擇資料庫，您的外部資料庫，然後檢視資料庫的詳細資訊。

如果您使用 Athena 建立和管理外部資料表，請使用 CREATE EXTERNAL SCHEMA 註冊資料庫。例如，下列命令可註冊名為 `sampledb` 的 Athena 資料庫。

```
create external schema athena_sample
from data catalog
database 'sampledb'
iam_role 'arn:aws:iam::123456789012:role/mySpectrumRole'
region 'us-east-1';
```

當您查詢 SVV_EXTERNAL_TABLES 系統檢視時，您可看到 Amazon Redshift `sampledb` 資料庫中的資料表，以及在 Amazon Redshift 中建立的資料表。

```
select * from svv_external_tables;
```

```
schemaname    | tablename        | location

--------------+------------------
+-----------------------------------------------------
athena_sample | elb_logs         | s3://athena-examples/elb/plaintext
athena_sample | lineitem_1t_csv  | s3://myspectrum/tpch/1000/lineitem_csv

athena_sample | lineitem_1t_part | s3://myspectrum/tpch/1000/lineitem_partition

spectrum      | sales            | s3://redshift-downloads/tickit/spectrum/sales

spectrum      | sales_part       | s3://redshift-downloads/tickit/spectrum/sales_part
```

註冊 Apache Hive 中繼存放區資料庫

如果您在 Apache Hive 中繼存放區中建立外部資料表，則可以使用 CREATE EXTERNAL SCHEMA 在 Redshift Spectrum 中註冊那些資料表。

在 CREATE EXTERNAL SCHEMA 陳述句中，指定 FROM HIVE METASTORE 子句並提供 Hive 中繼存放區 URI 和連接埠號碼。IAM 角色必須包含存取 Amazon S3 的許可，但不需要任何 Athena 許可。以下為註冊 Hive 中繼存放區的範例。

```
create external schema if not exists hive_schema
from hive metastore
database 'hive_database'
uri 'ip-10-0-111-111.us-west-2.compute.internal' port 9083
iam_role 'arn:aws:iam::123456789012:role/mySpectrumRole';
```

允許您的 Amazon Redshift 叢集存取 Amazon EMR 叢集

如果您的 Hive 中繼存放區在 Amazon EMR 中，您必須提供 Amazon Redshift 叢集對 Amazon EMR 叢集的存取權限。若要執行此作業，請建立 Amazon EC2 安全群組。您接著需要允許從您 Amazon Redshift 叢集的安全群組及您 Amazon EMR 叢集的安全群組，前往 EC2 安全群組的所有傳入流量。然後，將 EC2 安全性新增到 Amazon Redshift 叢集和 Amazon EMR 叢集。

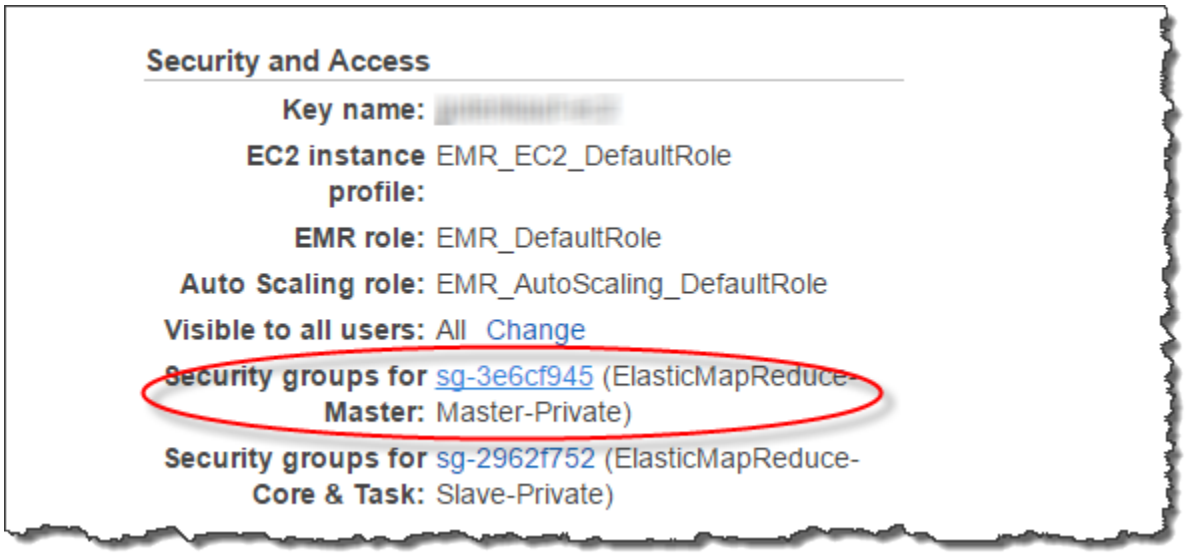
檢視您的 Amazon Redshift 叢集的安全群組名稱

如要顯示安全群組，請執行以下作業：

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後從清單選擇叢集以開啟其詳細資訊。
3. 選擇屬性，然後檢視網路與安全設定區段。
4. 在VPC 安全群組中找到您的安全群組並記下它。

檢視 Amazon EMR 主節點安全群組名稱

1. 開啟 Amazon EMR 叢集。如需詳細資訊，請參閱《Amazon EMR 管理指南》中的使用安全組態設定叢集安全性。
2. 在安全和存取下，記下 Amazon EMR 主節點安全群組名稱。



建立或修改 Amazon EC2 安全群組以允許 Amazon Redshift 和 Amazon EMR 之間的連線

1. 在 Amazon EC2 儀表板中，選擇安全群組。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的安全群組規則
2. 選擇建立安全群組。
3. 如果您使用的是 VPC，請選擇您的 Amazon Redshift 和 Amazon EMR 叢集所在的 VPC。
4. 新增傳入規則。
    1. 針對 Type (類型)，請選擇 Custom TCP (自訂 TCP)。
    2. 對於 Source (資源)，選擇 Custom (自訂)。

3. 輸入您 Amazon Redshift 安全群組的名稱。

5. 新增另一個傳入規則。

   1. 針對 Type (類型)，選擇 TCP。
   2. 針對 Port Range (連接埠範圍)，輸入 9083。

      > **Note**
      >
      > EMR HMS 的預設連接埠是 9083。如果您的 HMS 使用了不同連接埠，請在傳入規則和外部結構描述定義中指定該連接埠。

   3. 對於 Source (資源)，選擇 Custom (自訂)。

6. 輸入安全群組的名稱和說明。

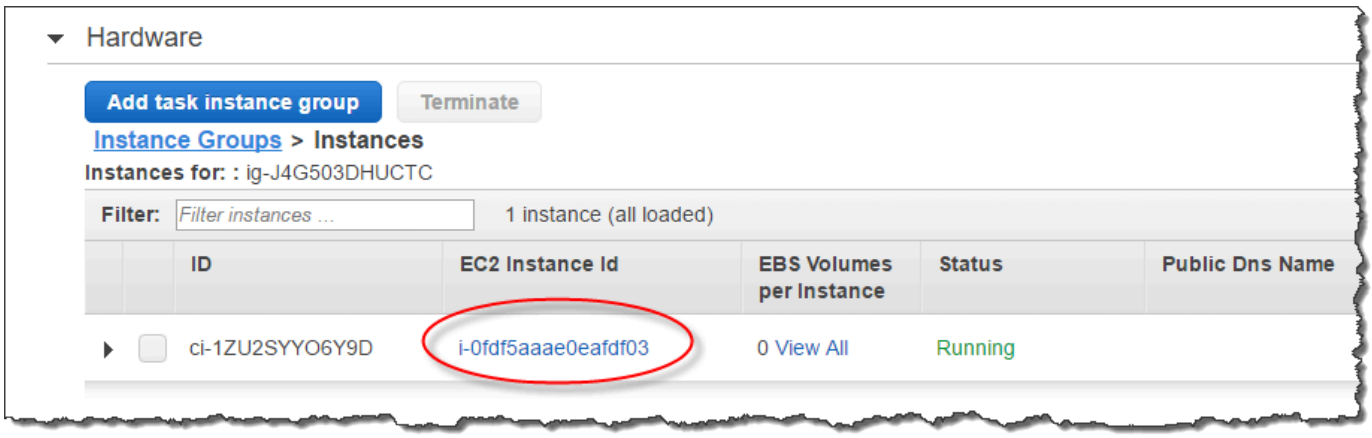
7. 選擇建立安全群組。

將您在先前程序中建立的 Amazon EC2 安全群組新增到 Amazon Redshift 叢集

1. 在 Amazon Redshift 中，選擇您的叢集。
2. 選擇 Properties (屬性)。
3. 檢視網路與安全設定，然後選擇編輯。
4. 在VPC 安全群組中，選擇新的安全群組名稱。
5. 選擇儲存變更。

將 Amazon EC2 安全群組新增到 Amazon EMR 叢集

1. 在 Amazon EMR 中，選擇您的叢集。如需詳細資訊，請參閱《Amazon EMR 管理指南》中的使用安全組態設定叢集安全性。
2. 在 Hardware (硬體) 下，選擇主節點的連結。
3. 選擇EC2 執行個體 ID欄中的連結。



4. 對於動作，選擇安全性、變更安全群組。
5. 在關聯的安全群組中，選擇新的安全群組，然後選擇新增安全群組。
6. 選擇儲存。

建立 Redshift Spectrum 外部資料表

您會在外部結構描述中建立外部資料表。您必須為外部結構描述的擁有者或超級使用者，始可建立外部資料表。若要轉移外部結構描述的所有權，請使用 ALTER SCHEMA 來變更擁有者。下列範例會將 spectrum_schema 結構描述的擁有者變更為 newowner。

```
alter schema spectrum_schema owner to newowner;
```

若要執行 Redshift Spectrum 查詢，您需要以下許可：

- 結構描述使用許可
- 在目前資料庫建立暫時資料表的許可

下列範例可在結構描述 spectrum_schema 上授予使用許可至 spectrumusers 使用者群組。

```
grant usage on schema spectrum_schema to group spectrumusers;
```

下列範例可在資料庫 spectrumdb 上授予臨時許可至 spectrumusers 使用者群組。

```
grant temp on database spectrumdb to group spectrumusers;
```

您可以在亞馬遜 Redshift， AWS Glue亞馬 Amazon Athena 或阿帕奇蜂巢中繼存儲中創建一個外部表。如需詳細資訊，請參閱《AWS Glue 開發人員指南》中的 AWS Glue使用入門、《Amazon Athena 使用者指南》中的 入門，或《Amazon EMR 開發人員指南》中的 Apache Hive。

如果您的外部資料表是在 Athena 或 Hive 中繼存放區中 AWS Glue定義的，您必須先建立參考外部資料庫的外部結構描述。然後，您可以透過在資料表名稱前加上結構描述名稱來參考 SELECT 陳述句中的外部資料表，且無需在 Amazon Redshift 中建立資料表。如需詳細資訊，請參閱 建立 Amazon Redshift Spectrum 外部結構描述。

若要允許 Amazon Redshift 查看中的表 AWS Glue Data Catalog，`glue:GetTable`請添加到 Amazon Redshift IAM 角色。否則，您可能會得到如下的錯誤：

```
RedshiftIamRoleSession is not authorized to perform: glue:GetTable on resource: *;
```

例如，假設您在 Athena 外部目錄中定義了名為 `lineitem_athena` 的外部資料表。在此狀況中，您可定義一個名為 `athena_schema` 的外部結構描述，然後使用下列 SELECT 陳述式來查詢資料表。

```
select count(*) from athena_schema.lineitem_athena;
```

若要在 Amazon Redshift 中定義外部資料表，請使用 CREATE EXTERNAL TABLE 命令。外部資料表陳述句定義了資料表欄、資料檔案格式，以及 Amazon S3 中的資料位置。Redshift Spectrum 會掃描指定資料夾以及任何子資料夾裡的檔案。Redshift Spectrum 會忽略以句號、底線或井號 (.、_ 或 #) 開頭，或以波狀符號 (~) 結尾的檔案和隱藏檔案。

以下範例在名為 `spectrum` 的 Amazon Redshift 外部結構描述中建立名為 SALES 的資料表。該資料位於 Tab 鍵分隔的文字檔案中。

```
create external table spectrum.sales(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
```

```
location 's3://redshift-downloads/tickit/spectrum/sales/'
table properties ('numRows'='172000');
```

若要檢視外部資料表，請查詢 SVV_EXTERNAL_TABLES 系統畫面。

虛擬資料欄

Amazon Redshift 預設會以虛擬資料欄 $path、$size 和 $spectrum_oid 建立外部資料表。選擇 $path 欄可檢視 Amazon S3 上資料檔案的路徑，選擇 $size 欄可檢視查詢傳回的每列資料檔案大小。$spectrum_oid 欄提供使用 Redshift Spectrum 執行關聯查詢的能力。如需範例，請參閱範例：在 Redshift Spectrum 中執行相互關聯子查詢。您必須以雙引號分隔 $path、$size 和 $spectrum_oid 欄名稱。SELECT * 子句不會傳回虛擬資料欄。您必須在查詢中明確包含 $path、$size 和 $spectrum_oid 欄名稱，如以下範例所示。

```
select "$path", "$size", "$spectrum_oid"
from spectrum.sales_part where saledate = '2008-12-01';
```

您可以藉由將 spectrum_enable_pseudo_columns 組態參數設定為 false，以停用工作階段的虛擬資料欄建立。如需詳細資訊，請參閱 spectrum_enable_pseudo_columns。您也可以將 enable_spectrum_oid 設定為 false，僅停用 $spectrum_oid 虛擬資料欄。如需詳細資訊，請參閱 enable_spectrum_oid。不過，停用 $spectrum_oid 虛擬資料欄也會停用對 Redshift Spectrum 相關查詢的支援。

> **Important**
>
> 選擇 $size、$path 或 $spectrum_oid 會產生費用，因為 Redshift Spectrum 會掃描 Amazon S3 上的資料檔案以判斷結果集的大小。如需詳細資訊，請參閱 Amazon Redshift 定價。

虛擬資料欄範例

下列範例會傳回外部資料表相關的資料檔案大小總和。

```
select distinct "$path", "$size"
from spectrum.sales_part;

 $path                                                                     | $size
---------------------------------------------------------------------------+-------
```

```
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/ |  1616
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/ |  1444
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/ |  1644
```

分割 Redshift Spectrum 外部資料表

當您分割資料時，可以在分割區索引鍵上進行篩選來限制 Redshift Spectrum 掃描的資料量。您可透過任何索引鍵來分割您的資料。

常見做法是根據時間對資料進行分割。例如，您可以選擇依年、月、日和小時來進行分割。如果您有來自多個來源的資料，則可以按資料來源識別碼和日期進行分割。

下列程序說明如何分割您的資料。

分割您的資料

1. 根據分割區索引鍵將資料儲存在 Amazon S3 的資料夾中。

   為每個分割區值建立一個資料夾，並使用分割區索引鍵和值命名該資料夾。例如，若您要依日期分割資料，您可能有名為 `saledate=2017-04-01`、`saledate=2017-04-02` 的資料夾，依此類推。Redshift Spectrum 會掃描分割區資料夾以及任何子資料夾裡的檔案。Redshift Spectrum 會忽略以句號、底線或井號 (.、_ 或 #) 開頭，或以波狀符號 (~) 結尾的檔案和隱藏檔案。
2. 建立外部資料表，並在 PARTITIONED BY 子句中指定分割區索引鍵。

   分割區索引鍵不得為資料表欄位的名稱。資料類型可能是 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE PRECISION、BOOLEAN、CHAR、VARCHAR、DATE、TIMESTAMP。
3. 新增分割區。

   使用 ALTER TABLE … ADD PARTITION，新增每個分割區，指定分割區欄與索引鍵值，以及 Amazon S3 中的分割區資料夾位置。您可以在單一 ALTER TABLE … ADD 陳述式中新增多個分割區。以下範例將為 `'2008-01'` 與 `'2008-03'` 新增分割區。

   ```
   alter table spectrum.sales_part add
   partition(saledate='2008-01-01')
   location 's3://redshift-downloads/tickit/spectrum/sales_partition/
   saledate=2008-01/'
   partition(saledate='2008-03-01')
   location 's3://redshift-downloads/tickit/spectrum/sales_partition/
   saledate=2008-03/';
   ```

> **Note**
>
> 如果您使用目 AWS Glue 錄，則可以使用單一 ALTER TABLE 陳述式新增最多 100 個分割區。

分割資料範例

在此範例中，您將建立一個由單一分割區索引鍵值分割的外部資料表，以及由兩個索引鍵值所分割的外部資料表。

此範例的範例資料位於 Amazon S3 儲存貯體中，可為所有經過驗證的 AWS 使用者提供讀取存取權限。您的叢集與外部資料檔案必須在相同的 AWS 區域。範例資料儲存貯體位於美國東部 (維吉尼亞北部) 區域 (us-east-1)。若要使用 Redshift Spectrum 存取資料，您的叢集必須也在 us-east-1 中。若要列以列出 Amazon S3 中的資料夾，請執行下列命令。

```
aws s3 ls s3://redshift-downloads/tickit/spectrum/sales_partition/
```

```
PRE saledate=2008-01/
PRE saledate=2008-03/
PRE saledate=2008-04/
PRE saledate=2008-05/
PRE saledate=2008-06/
PRE saledate=2008-12/
```

如果您沒有外部結構描述，請執行下列命令。將 Amazon 資源名稱 (ARN) 替換為您的 AWS Identity and Access Management (IAM) 角色。

```
create external schema spectrum
from data catalog
database 'spectrumdb'
iam_role 'arn:aws:iam::123456789012:role/myspectrumrole'
create external database if not exists;
```

範例 1：透過單一分割區索引鍵進行分割

在下列範例中，您會建立以月份進行分割的外部資料表。

若要建立以月份分割的外部資料表，請執行以下命令。

```
create external table spectrum.sales_part(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
partitioned by (saledate char(10))
row format delimited
fields terminated by '|'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales_partition/'
table properties ('numRows'='172000');
```

請執行下列 ALTER TABLE 命令以新增分割區。

```
alter table spectrum.sales_part add
partition(saledate='2008-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/'

partition(saledate='2008-03')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/'

partition(saledate='2008-04')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-04/';
```

若要從分割資料表選取資料，請執行下列查詢。

```
select top 5 spectrum.sales_part.eventid, sum(spectrum.sales_part.pricepaid)
from spectrum.sales_part, event
where spectrum.sales_part.eventid = event.eventid
  and spectrum.sales_part.pricepaid > 30
  and saledate = '2008-01'
group by spectrum.sales_part.eventid
order by 2 desc;
```

```
eventid | sum
```

```
--------+---------
   4124 | 21179.00
   1924 | 20569.00
   2294 | 18830.00
   2260 | 17669.00
   6032 | 17265.00
```

若要檢視外部資料表分割區，請查詢 SVV_EXTERNAL_PARTITIONS 系統畫面。

```
select schemaname, tablename, values, location from svv_external_partitions
where tablename = 'sales_part';
```

```
schemaname | tablename  | values      | location

-----------+------------+-------------
+-------------------------------------------------------------------------
spectrum   | sales_part | ["2008-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-01
spectrum   | sales_part | ["2008-03"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-03
spectrum   | sales_part | ["2008-04"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-04
```

範例 2：透過多個分割區索引鍵進行分割

若要建立以 date 與 eventid 分割的外部資料表，請執行以下命令。

```
create external table spectrum.sales_event(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
partitioned by (salesmonth char(10), event integer)
row format delimited
fields terminated by '|'
stored as textfile
```

```
location 's3://redshift-downloads/tickit/spectrum/salesevent/'
table properties ('numRows'='172000');
```

請執行下列 ALTER TABLE 命令以新增分割區。

```
alter table spectrum.sales_event add
partition(salesmonth='2008-01', event='101')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-01/
event=101/'

partition(salesmonth='2008-01', event='102')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-01/
event=102/'

partition(salesmonth='2008-01', event='103')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-01/
event=103/'

partition(salesmonth='2008-02', event='101')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-02/
event=101/'

partition(salesmonth='2008-02', event='102')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-02/
event=102/'

partition(salesmonth='2008-02', event='103')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-02/
event=103/'

partition(salesmonth='2008-03', event='101')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-03/
event=101/'

partition(salesmonth='2008-03', event='102')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-03/
event=102/'

partition(salesmonth='2008-03', event='103')
location 's3://redshift-downloads/tickit/spectrum/salesevent/salesmonth=2008-03/
event=103/';
```

執行下列查詢以從分割的資料表選擇資料。

```
select spectrum.sales_event.salesmonth, event.eventname,
 sum(spectrum.sales_event.pricepaid)
from spectrum.sales_event, event
where spectrum.sales_event.eventid = event.eventid
  and salesmonth = '2008-02'
 and (event = '101'
 or event = '102'
 or event = '103')
group by event.eventname, spectrum.sales_event.salesmonth
order by 3 desc;
```

```
salesmonth | eventname       | sum
-----------+-----------------+--------
2008-02    | The Magic Flute | 5062.00
2008-02    | La Sonnambula   | 3498.00
2008-02    | Die Walkure     |  534.00
```

將外部資料表資料欄映射到 ORC 資料欄

您會使用 Amazon Redshift Spectrum 外部資料表來查詢採用 ORC 格式的檔案中的資料。最佳化列單欄式 (ORC) 格式為單欄式儲存檔案格式，支援巢狀資料結構。如需關於查詢巢狀資料的相關資訊，請參閱使用 Amazon Redshift Spectrum 查詢巢狀資料。

當您建立參考 ORC 檔案中資料的外部資料表時，您會將外部資料表中的每個資料欄映射到 ORC 資料中的資料欄。若要這麼做，您會使用下列其中一個方法：

- 依位置映射
- 依資料欄名稱映射

依資料欄名稱映射為預設值。

依位置映射

利用位置映射，外部資料表中定義的第一個資料欄會映射到 ORC 資料檔案中的第一個資料欄，第二個映射到第二個，依此類推。依位置映射要求外部資料表中資料欄的順序與 ORC 檔案中的相符。如果資料欄的順序不相符，則可以依名稱映射資料欄。

Important

在舊版中，Redshift Spectrum 預設使用位置映射。如果您需要對現有資料表使用位置映射，請將資料表屬性 `orc.schema.resolution` 設為 `position`，如以下範例所示。

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='position');
```

例如，資料表 SPECTRUM.ORC_EXAMPLE 的定義如下。

```
create external table spectrum.orc_example(
int_col int,
float_col float,
nested_col struct<
  "int_col" : int,
  "map_col" : map<int, array<float >>
   >
) stored as orc
location 's3://example/orc/files/';
```

資料表結構的摘要如下。

```
• 'int_col' : int
• 'float_col' : float
• 'nested_col' : struct
   o 'int_col' : int
   o 'map_col' : map
      - key : int
      - value : array
         - value : float
```

基礎 ORC 檔案具有下列檔案結構。

```
• ORC file root(id = 0)
   o 'int_col' : int (id = 1)
   o 'float_col' : float (id = 2)
   o 'nested_col' : struct (id = 3)
      - 'int_col' : int (id = 4)
      - 'map_col' : map (id = 5)
```

```
         - key : int (id = 6)
         - value : array (id = 7)
            - value : float (id = 8)
```

在此範例中，您可以嚴格依位置將外部資料表中的每個資料欄映射至 ORC 檔案中的資料欄。以下顯示映射。

外部資料表資料欄名稱	ORC 資料欄 ID	ORC 資料欄名稱
int_col	1	int_col
float_col	2	float_col
nested_col	3	nested_col
nested_col.int_col	4	int_col
nested_col.map_col	5	map_col
nested_col.map_col.key	6	不適用
nested_col.map_col.value	7	不適用
nested_col.map_col.value.item	8	NA

依資料欄名稱映射

使用名稱映射，您可以將外部資料表中的資料欄映射至 ORC 檔案中相同層級、具有相同名稱的指定資料欄。

例如，假設您要將來自先前範例的資料表 SPECTRUM.ORC_EXAMPLE 與使用下列檔案結構的 ORC 檔案映射。

```
• ORC file root(id = 0)
   o 'nested_col' : struct (id = 1)
      - 'map_col' : map (id = 2)
         - key : int (id = 3)
         - value : array (id = 4)
            - value : float (id = 5)
      - 'int_col' : int (id = 6)
```

```
    o 'int_col' : int (id = 7)
    o 'float_col' : float (id = 8)
```

使用位置映射，Redshift Spectrum 會嘗試進行下列映射。

外部資料表資料欄名稱	ORC 資料欄 ID	ORC 資料欄名稱
int_col	1	struct
float_col	7	int_col
nested_col	8	float_col

查詢具有前述位置映射的資料表時，SELECT 命令會在類型驗證時失敗，因為結構不同。

您可以使用資料欄名稱映射，將相同的外部資料表映射至先前範例中所示的檔案結構。資料表資料欄 `int_col`、`float_col` 和 `nested_col` 會依資料欄名稱映射至 ORC 檔案中具有相同名稱的資料欄。外部資料表中名為 `nested_col` 的資料欄為 `struct` 資料欄，具有的子資料欄名為 `map_col` 和 `int_col`。子資料欄也會依資料欄名稱正確映射至 ORC 檔案中的對應資料欄。

為在 Apache Hudi 中管理的資料建立外部資料表

若要查詢 Apache Hudi Copy On Write (CoW) 格式的資料，您可以使用 Amazon Redshift Spectrum 外部資料表。Hudi Copy On Write 資料表是儲存在 Amazon S3 中的 Apache Parquet 檔案的集合。您可以讀取 Apache Hudi 版本 0.5.2、0.6.0、0.7.0、0.8.0、0.9.0、0.10.0、0.10.1、0.11.0 和 0.11.1 中的 Copy On Write (CoW) 資料表，這些表是透過插入、刪除和更新插入寫入操作來建立和修改的。例如，不支援 Bootstrap 資料表。如需詳細資訊，請參閱開放原始碼 Apache Hudi 文件中的 Copy On Write 資料表。

當您建立參考 Hudi CoW 格式之資料的外部資料表時，您會將外部資料表中的每個欄映射到 Hudi 資料中的欄。映射是透過欄完成的。

分割和未分割 Hudi 資料表的資料定義語言 (DDL) 陳述式類似於其他 Apache Parquet 檔案格式的 DDL 陳述式。對於 Hudi 資料表，您可以將 `INPUTFORMAT` 定義為 `org.apache.hudi.hadoop.HoodieParquetInputFormat`。`LOCATION` 參數必須指向包含 `.hoodie` 資料夾的 Hudi 資料表基底資料夾，這是建立 Hudi 提交時間表所必需的。在某些情況下，Hudi 資料表上的 SELECT 操作可能會失敗，並顯示找不到有效的 Hudi 提交時間表訊息。若是如此，請檢查 `.hoodie` 資料夾是否位於正確的位置並包含有效的 Hudi 提交時間表。

> **Note**
>
> 只有在使用 AWS Glue Data Catalog時才支援 Apache Hudi 格式。當您使用 Apache Hive 中繼存放區做為外部目錄時，不支援此功能。

定義未分割資料表的 DDL 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://s3-bucket/prefix'
```

定義分割資料表的 DDL 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
PARTITIONED BY(pcolumn1 pcolumn1-type[,...])
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://s3-bucket/prefix'
```

若要將分割區新增至分割的 Hudi 資料表，請執行 ALTER TABLE ADD PARTITION 命令，其中 LOCATION 參數指向 Amazon S3 子資料夾，其中包含屬於該分割區的檔案。

新增分割區的 DDL 具有下列格式。

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION 's3://s3-bucket/prefix/partition-path'
```

為在 Delta Lake 中管理的資料建立外部資料表

若要查詢 Delta Lake 資料表中的資料，您可以使用 Amazon Redshift Spectrum 外部資料表。

若要從 Redshift Spectrum 存取 Delta Lake 資料表，請在查詢前產生清單檔案。Delta Lake 清單檔案包含構成 Delta Lake 資料表一致快照的檔案清單。在分割資料表中，每個分割區都有一個清單檔

案。Delta Lake 資料表是儲存在 Amazon S3 中的 Apache Parquet 檔案的集合。如需詳細資訊，請參閱開放原始碼 Delta Lake 文件中的 Delta Lake。

當您建立參考 Delta Lake 資料表中資料的外部資料表時，您會將外部資料表中的每個欄映射到 Delta Lake 資料表中的欄。映射是透過欄名稱完成的。

分割和未分割 Delta Lake 資料表的 DDL 類似於其他 Apache Parquet 檔案格式的 DDL。對於 Delta Lake 資料表，您可以將 INPUTFORMAT 定義為 `org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat`，以及將 OUTPUTFORMAT 定義為 `org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat`。LOCATION 參數必須指向資料表基底資料夾中的清單檔案資料夾。如果 Delta Lake 資料表上的 SELECT 操作失敗，可能的原因請參閱Delta Lake 資料表的限制和疑難排解。

定義未分割資料表的 DDL 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://s3-bucket/prefix/_symlink_format_manifest'
```

定義分割資料表的 DDL 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
PARTITIONED BY(pcolumn1 pcolumn1-type[,...])
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://s3-bucket>/prefix/_symlink_format_manifest'
```

若要將分割區新增至分割的 Delta Lake 資料表，請執行 ALTER TABLE ADD PARTITION 命令，其中 LOCATION 參數指向包含分割區清單檔案的 Amazon S3 子資料夾。

新增分割區的 DDL 具有下列格式。

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION
's3://s3-bucket/prefix/_symlink_format_manifest/partition-path'
```

或者執行直接指向 Delta Lake 清單檔案的 DDL。

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION
's3://s3-bucket/prefix/_symlink_format_manifest/partition-path/manifest'
```

Delta Lake 資料表的限制和疑難排解

從 Redshift Spectrum 查詢 Delta Lake 資料表時，請考慮下列事項：

- 如果清單檔案指向不再存在的快照集或分割區，則查詢會失敗，直到產生新的有效清單檔案為止。例如，這可能是基礎資料表上的 VACUUM 操作所導致的，
- Delta Lake 清單檔案僅提供分割區層級一致性。

下表說明查詢 Delta Lake 資料表時出現某些錯誤的一些潛在原因。

錯誤訊息	可能原因
儲存貯體 s3-bucket-1 中的 Delta Lake 清單檔案不能包含儲存貯體 s3-bucket-2 中的項目。	清單檔案項目指向與指定儲存貯體不同的 Amazon S3 儲存貯體中的檔案。
Delta Lake 檔案預計會位於相同的資料夾中。	清單檔案項目指向具有與指定字首不同的 Amazon S3 字首的檔案。
找不到 Delta Lake 清單檔案 manifest-path 中列出的檔案 filename。	在 Amazon S3 中找不到清單檔案中列出的檔案。
擷取 Delta Lake 清單檔案時發生錯誤。	在 Amazon S3 中找不到清單檔案。
無效的 S3 路徑。	清單檔案檔案中的項目不是有效的 Amazon S3 路徑，或清單檔案檔案已損毀。

搭配 Amazon Redshift 使用 Apache Iceberg 資料表

您可以使用 Redshift Spectrum 或 Redshift Serverless 查詢 AWS Glue Data Catalog中編目的 Apache Iceberg 資料表。Apache Iceberg 是一種用於資料湖的開放原始碼資料表格式。如需詳細資訊，請參閱 Apache Iceberg 文件中的 Apache Iceberg。

Amazon Redshift 為查詢 Apache Iceberg 資料表提供交易一致性。您可以在使用 Amazon Redshift 執行查詢時，使用符合 ACID (原子性、一致性、隔離、耐久性) 的服務 (例如 Amazon Athena 和 Amazon EMR) 來操控資料表中的資料。Amazon Redshift 可以使用 Apache Iceberg 中繼資料中儲存的表格統計資料來最佳化查詢計劃並減少查詢處理期間的檔案掃描。使用 Amazon Redshift SQL，您可以將 Redshift 資料表與資料湖資料表結合在一起。

若要開始將 Iceberg 資料表與 Amazon Redshift 搭配使用：

1. 使用兼容的服務，如 Amazon 雅典娜或亞馬遜 EMR 在 AWS Glue Data Catalog 數據庫上創建 Apache 冰山表。若要使用 Athena 建立 Iceberg 資料表，請參閱《Amazon Athena 使用者指南》中的使用 Apache Iceberg 資料表。
2. 使用可存取資料湖的關聯 IAM 角色建立 Amazon Redshift 叢集或 Redshift Serverless 工作群組。如需如何建立叢集或工作群組的相關資訊，請參閱《Amazon Redshift 入門指南》中的 Amazon Redshift 佈建叢集和 Redshift Serverless。
3. 使用查詢編輯器 v2 或第三方 SQL 用戶端連線到您的叢集或工作群組。如需如何使用查詢編輯器 v2 連線的詳細資訊，請參閱亞馬遜 Redshift 管理指南中的使用 SQL 用戶端工具連線至 Amazon Redshift 資料倉儲。
4. 在 Amazon Redshift 資料庫中為包含 Iceberg 資料表的特定資料目錄資料庫建立外部結構描述。如需建立外部結構描述的相關資訊，請參閱建立 Amazon Redshift Spectrum 外部結構描述。
5. 執行 SQL 查詢以存取您建立的外部結構描述中的 Iceberg 資料表。

將 Apache Iceberg 資料表與 Amazon Redshift 搭配使用時的注意事項

將 Amazon Redshift 與 Iceberg 資料表結合使用時，請考慮以下事項：

- Iceberg 版本支援 - Amazon Redshift 支援對以下版本的 Iceberg 資料表執行查詢：
  - 版本 1 定義如何使用不可變資料檔案管理大型分析表。
  - 版本 2 增加支援資料列層級更新和刪除的功能，同時保持現有資料檔案不變，並使用刪除檔案來處理資料表資料變更。

  有關版本 1 和版本 2 資料表之間的區別，請參閱 Apache Iceberg 文件中的格式版本變更。
- 僅限查詢 - Amazon Redshift 支援 Apache Iceberg 資料表的唯讀存取。它支援交易一致的選擇查詢。您可以使用 Amazon Athena 等服務來定義和更新 AWS Glue Data Catalog中 Iceberg 資料表的結構描述。

- 新增分割區 - 您不需要為 Apache Iceberg 資料表手動新增分割區。Amazon Redshift 會自動偵測 Apache Iceberg 資料表中的新分割區，而且不需要手動操作即可更新資料表定義中的分割區。分割區規格中的任何變更也會自動套用至您的查詢，而無需任何使用者介入。
- 將 Iceberg 資料擷取至 Amazon Redshift - 您可以使用 INSERT INTO 或 CREATE TABLE AS 命令，將資料從 Iceberg 資料表匯入本機 Amazon Redshift 資料表。您目前無法使用 COPY 命令將 Apache Iceberg 資料表的內容擷取至本機 Amazon Redshift 資料表。
- 具體化視觀表 - 您可以在 Apache Iceberg 資料表上建立具體化視觀表，就像 Amazon Redshift 中的任何其他外部資料表一樣。其他資料湖資料表格式的考量相同，也適用於 Apache Iceberg 資料表。目前不支援資料湖資料表上的累加式更新、自動重新整理、自動查詢重新寫入和自動 MV。
- AWS Lake Formation 精細的存取控制 — Amazon Redshift 支援 Apache 冰山資料表上的 AWS Lake Formation 精細存取控制。
- 使用者定義的資料處理參數 - Amazon Redshift 支援 Apache Iceberg 資料表上的使用者定義資料處理參數。您可以在現有檔案上使用使用者定義的資料處理參數來自訂外部資料表中查詢的資料，以避免掃描錯誤。這些參數提供處理資料表結構描述與檔案實際資料之間不相符的功能。您也可以在 Apache Iceberg 資料表上使用使用者定義的資料處理參數。
- 資料共用 - Amazon Redshift 資料共用目前不支援資料湖資料表，包括 Apache Iceberg 資料表。
- 時間歷程查詢 - Apache Iceberg 資料表目前不支援時間歷程查詢。
- 定價 - 當您從叢集存取 Iceberg 資料表時，您需要支付 Redshift Spectrum 定價的費用。當您從工作群組存取 Iceberg 資料表時，您需要支付 Redshift Serverless 定價的費用。如需 Redshift Spectrum 和 Redshift Serverless 定價的相關資訊，請參閱 Amazon Redshift 定價。

主題

Apache Iceberg 資料表支援的資料類型

Amazon Redshift 可以查詢包含下列資料類型的 Iceberg 資料表：

```
binary
boolean
date
decimal
double
float
int
```

```
list
long
map
string
struct
timestamp without time zone
```

如需 Iceberg 資料類型的相關資訊，請參閱 Apache Iceberg 文件中的 Iceberg 結構描述。

下表顯示 Amazon Redshift 資料類型與 Iceberg 資料表資料類型之間的關係。

Iceberg 類型	Amazon Redshift 類型	備註
boolean	boolean	
-	tinyint	Amazon Redshift 中的 Iceberg 資料表不支援。
-	smallint	Amazon Redshift 中的 Iceberg 資料表不支援。
int	int	在 Amazon Redshift SQL 陳述式中，這種類型是 INTEGER。
long	bigint	
double	double	
float	float	
decimal(P, S)	decimal(P, S)	P 是總位數，S 是小數部分的位數。
-	char	Redshift Spectrum 中的 Iceberg 資料表不支援。
string	string	在 Amazon Redshift SQL 陳述式中，這種類型是 VARCHAR。
binary	binary	
date	date	
time	-	

Iceberg 類型	Amazon Redshift 類型	備註
`timestamp`	`timestamp`	
`timestamptz`	-	Redshift Spectrum 目前不支援 timestamptz 類型
`list<E>`	`array`	
`map<K,V>`	`map`	
`struct<...>`	`struct`	
`fixed(L)`	-	Redshift Spectrum 目前不支援 `fixed(L)` 類型。

如需 Amazon Redshift 中資料類型的相關資訊，請參閱資料類型。

改善 Amazon Redshift Spectrum 查詢效能

請查看查詢計畫以尋找已推送到 Amazon Redshift Spectrum 層的步驟。

下列步驟與 Redshift Spectrum 查詢相關：

- S3 循序掃描
- S3 HashAggregate
- S3 查詢掃描
- 序列掃描 PartitionInfo
- 分割區循環

以下範例顯示了將外部資料表與本地資料表連接的查詢的查詢計畫。請注意針對 Amazon S3 上的資料執行的 S3 序列掃描和 S3 HashAggregate 步驟。

```
explain
```

```
select top 10 spectrum.sales.eventid, sum(spectrum.sales.pricepaid)
from spectrum.sales, event
where spectrum.sales.eventid = event.eventid
and spectrum.sales.pricepaid > 30
group by spectrum.sales.eventid
order by 2 desc;
```

```
QUERY PLAN


------------------------------------------------------------------------------
XN Limit  (cost=1001055770628.63..1001055770628.65 rows=10 width=31)


  ->  XN Merge  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


        Merge Key: sum(sales.derived_col2)


        ->  XN Network  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


              Send to leader


              ->  XN Sort  (cost=1001055770628.63..1001055770629.13 rows=200 width=31)


                    Sort Key: sum(sales.derived_col2)


                    ->  XN HashAggregate  (cost=1055770620.49..1055770620.99 rows=200
 width=31)

                          ->  XN Hash Join DS_BCAST_INNER  (cost=3119.97..1055769620.49
 rows=200000 width=31)

                                Hash Cond: ("outer".derived_col1 = "inner".eventid)
```

```
                                ->  XN S3 Query Scan sales  (cost=3010.00..5010.50
rows=200000 width=31)

                                      ->  S3 HashAggregate  (cost=3010.00..3010.50
rows=200000 width=16)

                                            ->  S3 Seq Scan spectrum.sales
location:"s3://redshift-downloads/tickit/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)
                                                  Filter: (pricepaid > 30.00)


                                ->  XN Hash  (cost=87.98..87.98 rows=8798 width=4)


                                      ->  XN Seq Scan on event  (cost=0.00..87.98
rows=8798 width=4)
```

請注意查詢計畫中的以下元素：

- `S3 Seq Scan` 節點顯示的篩選條件 `pricepaid > 30.00` 已在 Redshift Spectrum 層中進行處理。

  `XN S3 Query Scan` 節點下的篩選條件節點指示了從 Redshift Spectrum 層傳回資料上方 Amazon Redshift 中的述詞處理。
- `S3 HashAggregate` 節點表示了 Redshift Spectrum 層中的 group by 子句 (`group by spectrum.sales.eventid`) 彙整。

以下為改善 Redshift Spectrum 效能的方式：

- 使用 Apache Parquet 格式的資料檔案。Parquet 會以單欄格式儲存資料，所以 Redshift Spectrum 可從掃描中消除不需要的欄位。當資料為文字檔案格式，Redshift Spectrum 將需要掃描整個檔案。
- 使用多個檔案以最佳化您的平行處裡。將您的檔案維持大於 64 MB。通過保持相同的檔案大小以避免資料大小扭曲。如需 Apache 實木複合地板檔案和設定建議的相關資訊，請參閱 Apache 文件中的檔案格式：組態。
- 在查詢中盡可能使用最少欄位。
- 將大型事實資料表放在 Amazon S3 中，並將常用的較小維度資料表保存在您本機 Amazon Redshift 資料庫中。

- 透過設定 TABLE PROPERTIES numRows 參數來更新外部資料表統計資訊。使用 CREATE EXTERNAL TABLE 或 ALTER TABLE 設定 TABLE PROPERTIES numRows 參數以反映資料表中的列數。Amazon Redshift 不會分析外部資料表來產生查詢最佳化工具用來產生查詢計劃的資料表統計資料。如果未設定外部資料表的資料表統計資料，則 Amazon Redshift 會產生查詢執行計畫。Amazon Redshift 根據以下假設產生此計劃：外部資料表為較大資料表，而本機資料表為較小資料表。
- Amazon Redshift 查詢計畫器會盡可能將述詞和彙總推送到 Redshift Spectrum 查詢層。當從 Amazon S3 傳回大量資料時，該處理將受到叢集資源的限制。Redshift Spectrum 將自動擴展以處理大量請求。因此，只要您可以將處理推送到 Redshift Spectrum 層，您的整體效能便會改善。
- 撰寫您的查詢以使用有資格推送到 Redshift Spectrum 層的篩選條件和彙總。

  以下是可以推送到 Redshift Spectrum 層的一些操作範例：

  - GROUP BY 子句
  - 比較條件和模式比對條件，例如 LIKE。
  - 彙總函數，例如 COUNT、SUM、AVG、MIN 與 MAX。
  - 字串函式。

  無法推送到 Redshift Spectrum 層的操作 (包括 DISTINCT 和 ORDER BY)。
- 使用分割區來限制掃描的資料。根據最常見的查詢述詞對資料進行分割，然後透過在分割區欄位進行篩選以刪除分割區。如需詳細資訊，請參閱 分割 Redshift Spectrum 外部資料表。

  查詢 SVL_S3PARTITION 以檢視分割區總數與合格的分割區。
- 使用 AWS Glue的統計資料產生器來計算資料表的資料行層級統計資 AWS Glue Data Catalog 料。一旦針對資料目錄中的表 AWS Glue 產生統計資料，Amazon Redshift Spectrum 就會自動使用這些統計資料來優化查詢計劃。如需有關使用計算資料行層級統計資料的詳細資訊 AWS Glue，請參閱AWS Glue 開發人員指南中的使用欄統計資料。

設定資料處理選項

您可以在建立外部資料表時設定資料表參數，以調整外部資料表中查詢的資料。否則，可能會發生掃描錯誤。如需詳細資訊，請參閱 CREATE EXTERNAL TABLE 中的 TABLE PROPERTIES。如需範例，請參閱 資料處理範例。如需錯誤清單，請參閱 SVL_SPECTRUM_SCAN_ERROR。

您可以在建立外部資料表時設定下列 TABLE PROPERTIES，以指定外部資料表中查詢之資料的輸入處理方式。

- column_count_mismatch_handling，識別檔案包含的列值是否少於或多於外部資料表定義中指定的欄數。
- invalid_char_handling，指定包含 VARCHAR、CHAR 和字串資料之欄中無效字元的輸入處理。當您為 invalid_char_handling 指定 REPLACE 時，您可以指定要使用的取代字元。
- numeric_overflow_handling，指定包含整數和小數資料之欄中的強制轉換溢位處理。
- surplus_bytes_handling，指定包含 VARBYTE 資料之欄中多餘位元組的輸入處理。
- surplus_char_handling，指定包含 VARCHAR、CHAR 和字串資料之欄中多餘字元的輸入處理。

您可以設定組態選項來取消超過錯誤數目上限的查詢。如需詳細資訊，請參閱 spectrum_query_maxerror。

範例：在 Redshift Spectrum 中執行相互關聯子查詢

您可以在 Redshift Spectrum 中執行相互關聯子查詢。$spectrum_oid 虛擬資料欄提供使用 Redshift Spectrum 執行相關查詢的能力。若要執行相互關聯子查詢，虛擬資料欄 $spectrum_oid 必須啟用，但不會出現在 SQL 陳述式中。如需詳細資訊，請參閱 虛擬資料欄。

若要為此範例建立外部結構描述和外部資料表，請參閱開始使用 Amazon Redshift Spectrum。

以下是 Redshift Spectrum 中的相互關聯子查詢範例。

```
select *
from myspectrum_schema.sales s
where exists
( select *
from myspectrum_schema.listing l
where l.listid = s.listid )
order by salesid
limit 5;
```

```
salesid  listid   sellerid   buyerid   eventid   dateid  qtysold  pricepaid
 commission   saletime
1        1        36861      21191     7872      1875    4        728         109.2
    2008-02-18 02:36:48
2        4        8117       11498     4337      1983    2        76          11.4
    2008-06-06 05:00:16
```

```
3       5       1616    17433   8647    1983    2       350     52.5
    2008-06-06 08:26:17
4       5       1616    19715   8647    1986    1       175     26.25
    2008-06-09 08:38:52
5       6       47402   14115   8240    2069    2       154     23.1
    2008-08-31 09:17:02
```

監控 Amazon Redshift Spectrum 中的指標

您可以使用以下的系統檢視來監控 Amazon Redshift Spectrum 查詢：

- SVL_S3QUERY

  使用 SVL_S3QUERY 畫面以取得有關區段和節點分割層級的 Redshift Spectrum 查詢 (S3 查詢) 詳細資訊。
- SVL_S3QUERY_SUMMARY

  使用 SVL_S3QUERY_SUMMARY 檢視以取得在系統上執行之所有 Amazon Redshift Spectrum 查詢 (S3 查詢) 摘要。

以下是 SVL_S3QUERY_SUMMARY 中要尋找的一些內容：

- Redshift Spectrum 查詢已處理的檔案數。
- 從 Amazon S3 掃描的位元組數。Redshift Spectrum 查詢的成本反映在從 Amazon S3 掃描的資料量中。
- 從 Redshift Spectrum 層傳回至叢集的位元組數。傳回的大量資料可能會影響系統效能。
- Redshift Spectrum 請求的最大持續期間和平均持續時間。長時間執行的請求可能表示存在瓶頸。

對 Amazon Redshift Spectrum 中的查詢進行疑難排解

您可以在以下內容中找到快速參考，了解和處理您在使用 Amazon Redshift Spectrum 查詢時可能遭遇的一些常見問題。查詢 SVL_S3LOG 系統資料表以檢視由 Redshift Spectrum 查詢產生的錯誤。

主題

- 超過重試次數
- 存取已限流

- 超過資源限制
- 分割的資料表沒有傳回資料列
- 未授權的錯誤
- 不相容的資料格式
- 在 Amazon Redshift 使用 Hive DDL 時的語法錯誤
- 建立暫存資料表的許可
- 無效的範圍
- 無效的 Parquet 版本編號

超過重試次數

如果 Amazon Redshift Spectrum 請求逾時，則該請求會被取消並重新提交。在五次重試失敗後，該查詢即失敗並顯示以下錯誤。

```
error:  Spectrum Scan Error: Retries exceeded
```

可能的子句包括：

- 大型檔案 (大於 1 GB)。檢查 Amazon S3 中的檔案大小，尋找大型檔案和檔案大小扭曲。將大型檔案分成許多介於 100 MB 與 1 GB 的較小檔案。試著將檔案維持相同大小。
- 慢速網路傳輸量。請稍後嘗試您的查詢。

存取已限流

Amazon Redshift Spectrum 受其他 AWS 服務的服務配額約束。在高使用量下，Redshift Spectrum 請求可能需要減慢速度，因而導致以下錯誤。

```
error:  Spectrum Scan Error: Access throttled
```

可能發生兩種類型的限流：

- 存取受到 Amazon S3 限流。
- 存取限制由. AWS KMS

錯誤內容提供有關限流類型的詳細資料。您可以在下面找到此限流的原因和可能的解決方案。

存取受到 Amazon S3 限流

如果字首上的讀取請求率太高，Amazon S3 可能會限流 Redshift Spectrum 請求。如需可在 Amazon S3 中實現的 GET/HEAD 請求率的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的最佳化 Amazon S3 效能。Amazon S3 GET/HEAD 請求率會將字首上的所有 GET/HEAD 請求列入考慮，因此存取相同字首的不同應用程式共用總請求率。

如果您的 Redshift Spectrum 請求經常受到 Amazon S3 的限流，請減少 Redshift Spectrum 對 Amazon S3 發出的 Amazon S3 GET/HEAD 請求數量。若要執行此動作，請嘗試將小型檔案合併成大型檔案。我們建議使用大於 64 MB 的檔案。

另外，請考慮分割 Redshift Spectrum 資料表，以從早期篩選中獲益，並減少 Amazon S3 中存取的檔案數量。如需詳細資訊，請參閱 分割 Redshift Spectrum 外部資料表。

存取由 AWS KMS限流

如果您使用伺服器端加密 (SSE-S3 或 SSE-KMS) 將資料存放在 Amazon S3，Amazon S3 會針對 Redshift 頻譜存取 AWS KMS 的每個檔案呼叫 API 操作。這些請求會計入您的密碼編譯操作配額；如需詳細資訊，請參閱 AWS KMS 請求配額。如需 SSE-S3 和 SSE-KMS 的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的使用伺服器端加密保護資料和搭配使用伺服器端加密與儲存在 AWS KMS中的金鑰來保護資料。

減少 Redshift Spectrum 發出的要求數目的第一個步驟 AWS KMS 是減少存取的檔案數目。若要執行此動作，請嘗試將小型檔案合併成大型檔案。我們建議使用大於 64 MB 的檔案。

如果您的 Redshift Spectrum 請求經常受到限制 AWS KMS，請考慮為加密操作的請求率要 AWS KMS 求增加配額。若要申請提高限制，請參閱Amazon Web Services 一般參考中的 AWS 服務限制。

超過資源限制

Redshift Spectrum 會對請求可以使用的記憶體數量強加上限。需要更多記憶體的 Redshift Spectrum 請求失敗，導致下列錯誤。

```
error:  Spectrum Scan Error: Resource limit exceeded
```

有兩個常見的原因可能會造成 Redshift Spectrum 請求超出其記憶體限額：

- Redshift Spectrum 會處理大塊資料，這些資料不能以較小的區塊分割。
- 大型彙總步驟是由 Redshift Spectrum 處理。

建議您使用支援平行讀取的檔案格式，分割大小為 128 MB 或更少。如需支援的檔案格式和建立資料檔的一般指導方針，請參閱在 Amazon Redshift Spectrum 為查詢建立資料檔案。使用不支援平行讀取的檔案格式或壓縮演算法時，建議您將檔案大小保持在 64 MB 與 128 MB 之間。

分割的資料表沒有傳回資料列

如果您的查詢沒有從分割的外部資料表傳回任何資料列，請檢查是否已為此外部資料表新增分割區。Redshift Spectrum 僅會掃描使用 ALTER TABLE … ADD PARTITION 明確新增之 Amazon S3 位置中的檔案。請查詢 SVV_EXTERNAL_PARTITIONS 畫面以尋找現有分割區。針對每個缺少的分割區執行 ALTER TABLE … ADD PARTITION。

未授權的錯誤

請確認叢集的 IAM 角色是否允許存取 Amazon S3 檔案物件。如果您的外部資料庫位於 Amazon Athena 上，請確認 IAM 角色是否允許存取 Athena 資源。如需詳細資訊，請參閱 Amazon Redshift Spectrum 的 IAM 政策。

不相容的資料格式

對於單欄檔案格式 (如 Apache Parquet)，該欄位類型已嵌入了資料。CREATE EXTERNAL TABLE 定義中的欄位類型必須與資料檔案的欄位類型相符。如果有不相符的部分，您將收到類似以下的錯誤訊息：

```
File 'https://s3bucket/location/file has an incompatible Parquet schema
for column ‘s3://s3bucket/location.col1'. Column type: VARCHAR, Par
```

由於訊息長度限制，錯誤訊息可能會被截斷。若要獲取包含欄位名稱與欄位類型的完整錯誤訊息，請查詢 SVL_S3LOG 系統畫面。

以下範例會查詢 SVL_S3LOG 以取得最後完成的查詢。

```
select message
from svl_s3log
where query = pg_last_query_id()
order by query,segment,slice;
```

以下是顯示完整錯誤訊息的結果範例。

```
                          message
```

```
----------------------------------------------------------------------------------------------
Spectrum Scan Error. File 'https://s3bucket/location/file has an incompatible
Parquet schema for column ' s3bucket/location.col1'.
Column type: VARCHAR, Parquet schema:\noptional int64 l_orderkey [i:0 d:1 r:0]\n
```

若要更正此錯誤，請修改外部資料表以符合 Parquet 檔案的欄位類型。

在 Amazon Redshift 使用 Hive DDL 時的語法錯誤

Amazon Redshift 支援 CREATE EXTERNAL TABLE 的資料定義語言 (DDL) (類似於 Hive DDL)。但是，這兩種類型的 DDL 有時並不完全相同。如果您複製了 Hive DDL 以建立或更改 Amazon Redshift 外部資料表，則可能會遇到語法錯誤。以下是 Amazon Redshift 與 Hive DDL 之間的差異範例：

- Amazon Redshift 需要單引號 (')，而 Hive DDL 支援雙引號 (")。
- Amazon Redshift 不支援 STRING 資料類型。請改用 VARCHAR。

建立暫存資料表的許可

要執行 Redshift Spectrum 查詢，資料庫使用者必須具有在資料庫中建立臨時資料表的許可。下列範例可在資料庫 `spectrumdb` 上授予臨時許可至 `spectrumusers` 使用者群組。

```
grant temp on database spectrumdb  to group spectrumusers;
```

如需詳細資訊，請參閱 GRANT。

無效的範圍

Redshift Spectrum 預期 Amazon S3 中屬於外部資料表的檔案在查詢期間不會被覆寫。如果發生這種情況，可能會導致以下錯誤。

```
Error: HTTP response error code: 416 Message: InvalidRange The requested range is not
 satisfiable
```

為避免發生錯誤，請確定 Amazon S3 檔案在使用 Redshift Spectrum 查詢時不會覆寫這些檔案。

無效的 Parquet 版本編號

Redshift Spectrum 會檢查它存取的每個 Apache Parquet 檔案的中繼資料。若檢查失敗，可能會導致類似下列內容的錯誤：

```
File 'https://s3.region.amazonaws.com/s3bucket/location/file has an invalid version
 number
```

導致檢查失敗的常見原因有兩個：

- Parquet 檔案在查詢期間已被覆寫 (請參閱無效的範圍)。
- Parquet 檔案已損毀。

教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料

概要

Amazon Redshift Spectrum 支援查詢 Parquet、ORC、JSON 和 Ion 檔案格式的巢狀資料。Redshift Spectrum 會存取使用外部資料表的資料。您可以建立使用複雜資料類型 `struct`、`array` 和 `map` 的外部資料表。

例如，假設您的資料檔案在 Amazon S3 名為 `customers` 的資料夾中包含下列資料。雖然沒有單一根元素，此取樣資料中的每個 JSON 物件代表資料表中的資料列。

```
{"id": 1,
 "name": {"given": "John", "family": "Smith"},
 "phones": ["123-457789"],
 "orders": [{"shipdate": "2018-03-01T11:59:59.000Z", "price": 100.50},
            {"shipdate": "2018-03-01T09:10:00.000Z", "price": 99.12}]
}
{"id": 2,
 "name": {"given": "Jenny", "family": "Doe"},
 "phones": ["858-8675309", "415-9876543"],
 "orders": []
}
{"id": 3,
 "name": {"given": "Andy", "family": "Jones"},
 "phones": [],
 "orders": [{"shipdate": "2018-03-02T08:02:15.000Z", "price": 13.50}]
}
```

您現在可以使用 Amazon Redshift Spectrum 來查詢檔案中的巢狀資料。下列教學課程說明如何使用 Apache Parquet 資料執行此操作。

如需教學課程必要條件、步驟和巢狀資料的使用案例，請參閱下列主題：

必要條件

如果您尚未使用 Redshift Spectrum，請先遵循開始使用 Amazon Redshift Spectrum中的步驟，再繼續進行。

如要建立外部結構描述，請將以下命令中的 IAM 角色 ARN 替換為您在建立 IAM 角色中建立的角色 ARN。然後，在您的 SQL 用戶端中執行命令。

```
create external schema spectrum
from data catalog
database 'myspectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
create external database if not exists;
```

步驟 1：建立包含巢狀資料的外部資料表

您可以從 Amazon S3 下載來源資料以檢視來源資料。

若要建立此教學課程的外部資料表，請執行以下命令。

```
CREATE EXTERNAL TABLE spectrum.customers (
  id     int,
  name   struct<given:varchar(20), family:varchar(20)>,
  phones array<varchar(20)>,
  orders array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/tickit/spectrum/customers/';
```

在上述範例中，外部資料表 `spectrum.customers` 使用 `struct` 和 `array` 資料類型來定義具有巢狀資料的資料欄。Amazon Redshift Spectrum 支援查詢 Parquet、ORC、JSON 和 Ion 檔案格式的巢

狀資料。Apache Parquet 檔案的 STORED AS 參數是 PARQUET。LOCATION 參數必須參考包含巢狀資料或檔案的 Amazon S3 資料夾。如需詳細資訊，請參閱CREATE EXTERNAL TABLE。

您可以在任何層級將 array 和 struct 類型形成巢狀。例如，您可以如以下範例所示定義名為 toparray 的資料欄。

```
toparray array<struct<nestedarray:
         array<struct<morenestedarray:
         array<string>>>>>
```

您也可以如以下範例中的資料欄 x 所示，將 struct 類型形成巢狀。

```
x struct<a: string,
         b: struct<c: integer,
                   d: struct<e: string>
                  >
        >
```

步驟 2：在 Amazon S3 中使用 SQL 延伸模組查詢您的巢狀資料

Redshift Spectrum 支援透過 Amazon Redshift SQL 語法的延組模組查詢 array、map 和 struct 複雜類型。

延組模組 1：存取 structs 的欄位

您可以使用可將欄位名稱串連為路徑的點表示法，從 struct 資料欄擷取資料。例如，以下查詢會傳回客戶的名字和姓氏。名字是透過長路徑 c.name.given 存取。姓氏是透過長路徑 c.name.family 存取。

```
SELECT c.id, c.name.given, c.name.family
FROM   spectrum.customers c;
```

前述查詢會傳回下列資料。

```
id | given | family
---|-------|-------
1  | John  | Smith
2  | Jenny | Doe
3  | Andy  | Jones
```

```
(3 rows)
```

struct 可以是另一個 struct 的資料欄，可以是任何層級的另一個 struct 的資料欄。存取這類深入巢狀 struct 中資料欄的路徑可以有任意長度。例如，查看以下範例中資料欄 x 的定義。

```
x struct<a: string,
         b: struct<c: integer,
                   d: struct<e: string>
                  >
        >
```

您可以以 x.b.d.e 的形式存取 e 中的資料。

延伸模組 2：在 FROM 子句中存取 array 的範圍

您可以從 array 資料欄擷取資料 (並且透過延伸模組 map 資料欄)，方法是在 FROM 子句中指定 array 資料欄以取代資料表名稱。延伸模組會套用到主查詢的 FROM 子句，以及子查詢的 FROM 子句。

您可以依位置參考 array 元素，例如 c.orders[0]。(預覽)

藉由將範圍與 arrays 聯結結合，您可以達成各種解巢狀，如下列使用案例所述。

使用內部聯結解巢狀

下列查詢會選取具有訂單之客戶的客戶 ID 和訂單出貨日期。FROM 子句 c.orders o 中的 SQL 延伸模組取決於別名 c。

```
SELECT c.id, o.shipdate
FROM   spectrum.customers c, c.orders o
```

針對具有訂單的每個客戶 c，FROM 子句會為客戶 c 的每個訂單 o 傳回一個資料列。該資料列會結合客戶資料列 c 和訂單資料列 o。那麼，SELECT 子句只會保留 c.id 和 o.shipdate。結果如下所示。

```
id|      shipdate
--|---------------------
1 |2018-03-01  11:59:59
1 |2018-03-01  09:10:00
3 |2018-03-02  08:02:15
(3 rows)
```

別名 c 提供客戶欄位的存取，而別名 o 則提供訂單欄位的存取。

語意類似於標準的 SQL。您可以將 FROM 子句想成執行下列巢狀迴圈，它的後面接著 SELECT 選擇要輸出的欄位。

```
for each customer c in spectrum.customers
  for each order o in c.orders
     output c.id and o.shipdate
```

因此，如果客戶沒有訂單，客戶便不會出現在結果中。

您也可以將這想成執行 JOIN 搭配 customers資料表和 orders 陣列的 FROM 子句。實際上，您也可以如下列範例所示寫入查詢。

```
SELECT c.id, o.shipdate
FROM   spectrum.customers c INNER JOIN c.orders o ON true
```

> **Note**
>
> 如果名為 c 的結構描述存在於名為 orders 的資料表中，則 c.orders 會參考資料表 orders，而非 customers 的陣列資料欄。

使用左聯結解巢狀

下列查詢會輸出所有客戶名稱和其訂單。如果客戶尚未下訂單，仍會傳回客戶的名稱。不過，在此情況下，訂單資料欄會是 NULL，如以下 Jenny Doe 的範例所示。

```
SELECT c.id, c.name.given, c.name.family, o.shipdate, o.price
FROM   spectrum.customers c LEFT JOIN c.orders o ON true
```

前述查詢會傳回下列資料。

```
id  |  given  | family  |    shipdate          | price
----|---------|---------|----------------------|--------
 1  |  John   | Smith   | 2018-03-01  11:59:59 | 100.5
 1  |  John   | Smith   | 2018-03-01  09:10:00 |  99.12
 2  |  Jenny  | Doe     |                      |
 3  |  Andy   | Jones   | 2018-03-02  08:02:15 |  13.5
 (4 rows)
```

延伸模組 3：直接使用別名存取純量的陣列

當 FROM 子句中別名 p 的範圍超過純量的陣列，則查詢會完全以 p 的形式參考 p 的值。例如，以下查詢會產生客戶名稱與電話號碼的配對。

```
SELECT c.name.given, c.name.family, p AS phone
FROM   spectrum.customers c LEFT JOIN c.phones p ON true
```

前述查詢會傳回下列資料。

```
given  |  family  |  phone
-------|----------|-----------
John   | Smith    | 123-4577891
Jenny  | Doe      | 858-8675309
Jenny  | Doe      | 415-9876543
Andy   | Jones    |
(4 rows)
```

延伸模組 4：存取 map 的元素

Redshift Spectrum 會將 map 資料類型視為 array 類型，其中包含的 struct 類型具有 key 資料欄和 value 資料欄。key 必須是 scalar；值可以是任何資料類型。

例如，以下程式碼會建立外部資料表，其中具有 map 用於儲存電話號碼。

```
CREATE EXTERNAL TABLE spectrum.customers2 (
  id     int,
  name   struct<given:varchar(20), family:varchar(20)>,
  phones map<varchar(20), varchar(20)>,
  orders array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/tickit/spectrum/customers/';
```

因為 map 類型的行為類似具有資料欄 key 和 value 的 array 類型，您可以將前述結構描述想像成如下。

```
CREATE EXTERNAL TABLE spectrum.customers3 (
  id     int,
  name   struct<given:varchar(20), family:varchar(20)>,
  phones array<struct<key:varchar(20), value:varchar(20)>>,
```

```
  orders array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/tickit/spectrum/customers/';
```

下列查詢會傳回具有行動電話號碼之客戶的名稱，並傳回每個名稱的號碼。映射查詢被視為與查詢 struct 類型的巢狀 array 的同等項目。下列查詢只會在您如先前所述建立了外部資料表時傳回資料。

```
SELECT c.name.given, c.name.family, p.value
FROM   spectrum.customers c, c.phones p
WHERE  p.key = 'mobile';
```

> **Note**
>
> map 的 key 為用於 Ion 和 JSON 檔案類型的 string。

巢狀資料使用案例

您可以將前述的延伸模組與一般的 SQL 功能結合。以下使用案例說明一些常見的組合。這些範例可協助示範如何使用巢狀資料。本教學課程中未提供。

主題

- 擷取巢狀資料
- 使用子查詢彙整巢狀資料
- 聯結 Amazon Redshift 和巢狀資料

擷取巢狀資料

您可以使用 CREATE TABLE AS 陳述式從包含複雜資料類型的外部資料表擷取資料。下列查詢會使用 LEFT JOIN 從外部資料表擷取所有客戶和其電話號碼，並將該資訊存放在 Amazon Redshift 資料表 CustomerPhones。

```
CREATE TABLE CustomerPhones AS
SELECT  c.name.given, c.name.family, p AS phone
FROM    spectrum.customers c LEFT JOIN c.phones p ON true;
```

使用子查詢彙整巢狀資料

您也可以使用子查詢來彙總巢狀資料。以下範例說明此方法。

```
SELECT c.name.given, c.name.family, (SELECT COUNT(*) FROM c.orders o) AS ordercount
FROM   spectrum.customers c;
```

會傳回下列資料。

```
given   |  family  |  ordercount
--------|----------|--------------
 Jenny  |  Doe     |       0
 John   |  Smith   |       2
 Andy   |  Jones   |       1
 (3 rows)
```

> **Note**
>
> 依上層資料列群組來彙總巢狀資料時，最有效的方式是先前範例中說明的方式。在該範例中，c.orders 的巢狀資料列會依其上層資料列 c 群組。或者，如果您知道每個 customer 的 id 是唯一的，並且 o.shipdate 絕不會是 null，則可以如以下範例所示彙總。不過，此方法一般不如先前的範例來得有效。

```
SELECT    c.name.given, c.name.family, COUNT(o.shipdate) AS ordercount
FROM      spectrum.customers c LEFT JOIN c.orders o ON true
GROUP BY  c.id, c.name.given, c.name.family;
```

您也可以在參照上階查詢別名 (c) 的 FROM 子句中使用子查詢來編寫查詢，並擷取陣列資料。下列範例示範此方法。

```
SELECT c.name.given, c.name.family, s.count AS ordercount
FROM   spectrum.customers c, (SELECT count(*) AS count FROM c.orders o) s;
```

聯結 Amazon Redshift 和巢狀資料

您也可以在外部資料表中聯結 Amazon Redshift 資料和巢狀資料。例如，假設您在 Amazon S3 中有以下巢狀資料。

```
CREATE EXTERNAL TABLE spectrum.customers2 (
  id      int,
  name    struct<given:varchar(20), family:varchar(20)>,
  phones  array<varchar(20)>,
  orders  array<struct<shipdate:timestamp, item:int>>
);
```

同時假設您在 Amazon Redshift 中有以下資料表。

```
CREATE TABLE prices (
  id int,
  price double precision
);
```

下列查詢會基於上述尋找每個客戶購買的總數量和金額。下列範例僅為示範。它只會在您如先前所述建立了資料表時傳回資料。

```
SELECT   c.name.given, c.name.family, COUNT(o.date) AS ordercount, SUM(p.price) AS
 ordersum
FROM     spectrum.customers2 c, c.orders o, prices p ON o.item = p.id
GROUP BY c.id, c.name.given, c.name.family;
```

巢狀資料限制 (預覽)

> **Note**
>
> 以下清單中標記 (預覽) 的限制僅適用於預覽叢集，以及在下列區域中建立的預覽工作群組。
>
> - 美國東部 (俄亥俄) (us-east-2)
> - 美國東部 (維吉尼亞北部) (us-east-1)
> - 美國西部 (加利佛尼亞北部) (us-west-1)
> - 亞太區域 (東京) (ap-northeast-1)
> - 歐洲 (愛爾蘭) (eu-west-1)
> - 歐洲 (斯德哥爾摩) (eu-north-1)
>
> 如需有關設定預覽叢集的詳細資訊，請參閱 Amazon Redshift 管理指南中的建立預覽叢集。如需設定預覽工作群組的詳細資訊，請參閱 Amazon Redshift 管理指南中的建立預覽工作群組。

下列限制適用巢狀資料：

- array 或 map 類型可以包含其他 array 或 map 類型，只要嵌套 arrays 上的查詢或 maps 不傳回 scalar 值。(預覽)
- Amazon Redshift Spectrum 僅以外部資料表形式支援複雜資料類型。
- 子查詢結果資料欄必須是最上層。(預覽)
- 如果 OUTER JOIN 表達式參照巢狀資料表，則只能參考該資料表和其巢狀陣列 (和對映)。如果 OUTER JOIN 表達式未參考巢狀資料表，則可以參考任何數量的非巢狀資料表。
- 如果子查詢中的 FROM 子句參考巢狀資料表，則不能參考任何其他資料表。
- 如果子查詢取決於參考上層資料表的巢狀資料表，則子查詢只能在 FROM 子句中使用上層資料表。您無法在任何其他子句中使用上層，例如 SELECT 或 WHERE 子句。例如，下列查詢不會執行，因為子查詢的 SELECT 子句參照上層資料表 c。

```
SELECT c.name.given
FROM   spectrum.customers c
WHERE (SELECT COUNT(c.id) FROM c.phones p WHERE p LIKE '858%') > 1;
```

下列查詢可運作，因為上層 c 僅用於子查詢的 FROM 子句。

```
SELECT c.name.given
FROM   spectrum.customers c
WHERE (SELECT COUNT(*) FROM c.phones p WHERE p LIKE '858%') > 1;
```

- 存取 FROM 子句以外位置巢狀資料的子查詢必須傳回單一值。唯一的例外是 WHERE 子句中的 (NOT) EXISTS 運算子。
- 不支援 (NOT) IN。
- 所有巢狀類型的最大深度為 100。此限制適用所有檔案格式 (Parquet、ORC、Ion 和 JSON)。
- 存取巢狀資料的彙總子查詢只能參考其 FROM 子句中 (而非外部資料表) 的 arrays 和 maps。
- 不支援查詢 Redshift Spectrum 資料表中巢狀資料的虛擬資料欄。如需詳細資訊，請參閱虛擬資料欄。
- 當透過在 FROM 子句中指定陣列或映射資料欄來擷取資料時，您只能從這些資料欄中選擇值 (如果值為 scalar)。例如，下列查詢會嘗試從陣列內部的 SELECT 元素。選取 arr.a 的查詢是有效的，因為 arr.a 是一個 scalar 值。第二個查詢不起作用，因為 array 是從 FROM 子句中的 s3.nested table 中擷取的陣列。(預覽)

```
SELECT array_column FROM s3.nested_table;
```

```
array_column
-----------------
[{"a":1},{"b":2}]

SELECT arr.a FROM s3.nested_table t, t.array_column arr;

arr.a
-----
1

--This query fails to run.
SELECT array FROM s3.nested_table tab, tab.array_column array;
```

您不能在本身來自另一個陣列或映射的 FROM 子句中使用陣列或映射。若要選取陣列或嵌套在其他陣列內的其他複雜結構，請考慮在 SELECT 陳述式中使用索引。

序列化複雜的巢狀 JSON

本教學課程中示範方法的替代方法是以序列化 JSON 形式查詢頂層巢狀集合欄。您可以透過 Redshift Spectrum 使用序列化來檢查、轉換巢狀資料並將其提取為 JSON。ORC、JSON、Ion 和 Parquet 格式支援此方法。使用工作階段組態參數 `json_serialization_enable` 來設定序列化行為。設定時，複雜的 JSON 資料類型會序列化為 VARCHAR(65535)。您可以使用 JSON 函數 存取巢狀 JSON。如需詳細資訊，請參閱json_serialization_enable。

例如，如果沒有設定 `json_serialization_enable`，下列直接存取巢狀欄的查詢會失敗。

```
SELECT * FROM spectrum.customers LIMIT 1;

=> ERROR:  Nested tables do not support '*' in the SELECT clause.

SELECT name FROM spectrum.customers LIMIT 1;

=> ERROR:  column "name" does not exist in customers
```

設定 `json_serialization_enable` 可讓您直接查詢頂層集合。

```
SET json_serialization_enable TO true;

SELECT * FROM spectrum.customers order by id LIMIT 1;
```

```
id | name                                  | phones         | orders
---+---------------------------------------+----------------
+-----------------------------------------------------------------------------------------------------------------
1  | {"given": "John", "family": "Smith"} | ["123-457789"] | [{"shipdate":
 "2018-03-01T11:59:59.000Z", "price": 100.50}, {"shipdate": "2018-03-01T09:10:00.000Z",
 "price": 99.12}]

SELECT name FROM spectrum.customers order by id LIMIT 1;

name
---------
{"given": "John", "family": "Smith"}
```

序列化巢狀 JSON 時，請考慮下列項目。

- 當集合欄序列化為 VARCHAR(65535) 時，無法直接存取其巢狀子欄位做為查詢語法的一部分 (例如，在篩選條件子句中)。但是，JSON 函數可用於存取巢狀 JSON。
- 不支援下列特殊表示法：
  - ORC 聯集
  - 具有複雜類型鍵的 ORC 映射
  - Ion 資料包
  - Ion SEXP
- 時間戳記會傳回為 ISO 序列化字串。
- 原始映射鍵被提升為字串 (例如，1 到 "1")。
- 頂層 null 值會序列化為 NULL。
- 如果序列化超出最大 VARCHAR 的大小 65535，則儲存格將設為 NULL。

序列化包含 JSON 字串的複雜類型

預設情況下，巢狀集合中包含的字串值將序列化為逸出的 JSON 字串。當字串是有效的 JSON 時，逸出可能是不可取的。相反，您可能希望將 VARCHAR 的巢狀子元素或欄位直接編寫為 JSON。使用 `json_serialization_parse_nested_strings` 工作階段層級組態啟用此行為。同時設定 `json_serialization_enable` 和 `json_serialization_parse_nested_strings` 時，有效的 JSON 值會以內嵌方式序列化，不含逸出字元。當值不是有效的 JSON 時，會將其逸出，就好像未設定 `json_serialization_parse_nested_strings` 組態值一樣。如需詳細資訊，請參閱json_serialization_parse_nested_strings。

例如，假設上一個範例中的資料在 name VARCHAR(20) 欄位中包含 JSON 作為 structs 複雜類型：

```
name
---------
{"given": "{\"first\":\"John\",\"middle\":\"James\"}", "family": "Smith"}
```

設定 json_serialization_parse_nested_strings 時，name 欄會依照下列方式序列化：

```
SET json_serialization_enable TO true;
SET json_serialization_parse_nested_strings TO true;
SELECT name FROM spectrum.customers order by id LIMIT 1;

name
---------
{"given": {"first":"John","middle":"James"}, "family": "Smith"}
```

而不是這樣逸出：

```
SET json_serialization_enable TO true;
SELECT name FROM spectrum.customers order by id LIMIT 1;

name
---------
{"given": "{\"first\":\"John\",\"middle\":\"James\"}", "family": "Smith"}
```

在 Amazon Redshift 中使用 HyperLogLog 草圖

HyperLog日誌是用於估計一個多集的基數的算法。基數是指多重集合中相異值的數目。例如，在 {4,3,6,2,2,6,4,3,6,2,2,3} 的集合中，基數為 4，因為不同的值有 4、3、6 和 2。

HyperLogLog 演算法的精確度 (也稱為 m 值) 會影響估計基數的準確度。在基數估算期間，Amazon Redshift 會使用預設精確度值 15。針對較小的資料集，此值最多可以到 26。因此，平均相對誤差範圍介於 0.01–0.6% 之間。

計算多集的基數時， HyperLogLog 演算法會產生一個稱為 HLL 草圖的結構。HLL 草圖會封裝多重集合中不同值的相關資訊。Amazon Redshift 資料類型 HLLSKETCH 代表這類草圖值。此資料類型可用來將草圖儲存在 Amazon Redshift 資料表中。此外，Amazon Redshift 支援可套用至 HLLSKETCH 值來作為彙總和純量函數的操作。您可以使用這些函數來擷取 HLLSKETCH 的基數，並組合多個 HLLSKETCH 值。

從大型資料集擷取基數時，HLLSKETCH 資料類型可提供顯著的查詢效能優勢。您可以使用 HLLSKETCH 值預先彙總這些資料集，並將其儲存在資料表中。Amazon Redshift 可以直接從儲存的 HLLSKETCH 值擷取基數，而無需存取基礎資料集。

在處理 HLL 草圖時，Amazon Redshift 會執行最佳化，以最大程度地減少草圖的記憶體佔用量，並最大限度地提高所擷取基數的精確度。Amazon Redshift 會針對 HLL 草圖使用兩種表示法：稀疏和密集。HLLSKETCH 會以稀疏格式開始。當新值插入到其中時，其大小會增加。當其大小達到密集表示的大小後，Amazon Redshift 會自動將草圖從稀疏轉換為密集。

當草圖為稀疏格式時，Amazon Redshift 會將 HLLSKETCH 匯入、匯出和列印為 JSON。當草圖為密集格式時，Amazon Redshift 會將 HLLSKETCH 匯入、匯出和列印為 Base64 字串。如需 UNLOAD 的相關資訊，請參閱 卸載 HLLSKETCH 資料類型。若要將文字或逗號分隔值 (CSV) 資料匯入 Amazon Redshift，請使用 COPY 命令。如需詳細資訊，請參閱 載入 HLLSKETCH 資料類型。

如需搭配使用之函數的資訊 HyperLogLog，請參閱HyperLogLog 函數。

主題

考量事項

以下是 HyperLogLog 在 Amazon Redshift 中使用的注意事項：

- 下列非HyperLogLog 函數可以接受 HLLSKETCH 類型的輸入或 HLLSKETCH 類型的欄：
  - COUNT 彙整函數
  - COALESCE 和 NVL 條件表達式
  - CASE 表達式
- 支援的編碼為 RAW。
- 您可以在含有 HLLSKETCH 資料欄的資料表上執行 UNLOAD 操作，並將其轉換為文字或 CSV。您可以使用 UNLOAD HLLSKETCH 資料欄來寫入 HLLSKETCH 資料。Amazon Redshift 會針對稀疏表示使用 JSON 格式顯示資料，或針對密集表示使用 Base64 格式顯示資料。如需 UNLOAD 的相關資訊，請參閱 卸載 HLLSKETCH 資料類型。

  以下顯示用於以 JSON 格式表示的稀疏 HyperLogLog 草圖的格式。

  ```
  {"version":1,"logm":15,"sparse":{"indices":
  [15099259,33107846,37891580,50065963],"values":[2,3,2,1]}}
  ```
- 您可以使用COPY 命令將文字或 CSV 資料匯入 Amazon Redshift。如需詳細資訊，請參閱 載入 HLLSKETCH 資料類型。
- HLLSKETCH 的預設編碼為 RAW。如需詳細資訊，請參閱 壓縮編碼。

限制

以下是 HyperLogLog 在 Amazon Redshift 中使用的限制：

- Amazon Redshift 資料表不支援 HLLSKETCH 資料欄作為 Amazon Redshift 資料表的排序索引鍵或分佈索引鍵。
- Amazon Redshift 不支援在 ORDER BY、GROUP BY 或 DISTINCT 子句中使用 HLLSKETCH 資料欄。
- 您只能將 HLLSKETCH 資料欄 UNLOAD 至文字或 CSV 格式。然後，Amazon Redshift 會以 JSON 格式或 Base64 格式寫入 HLLSKETCH 資料。如需 UNLOAD 的相關資訊，請參閱 UNLOAD。
- Amazon Redshift 僅支援精確度為 15 的 HyperLogLog 草圖。
- JDBC 和 ODBC 驅動程式不支援 HLLSKETCH 資料類型。因此，結果集會使用 VARCHAR 來表示 HLLSKETCH 值。

- Amazon Redshift Spectrum 本身不支援 HLLSKETCH 資料。因此，您無法建立或改變包含 HLLSKETCH 資料欄的外部資料表。
- Python 使用者定義函數 (UDF) 的資料類型不支援 HLLSKETCH 資料類型。如需 Python UDF 的相關資訊，請參閱 建立純量 Python UDF。

範例

範例：在子查詢中傳回基數

下列範例會針對名為 Sales 的資料表，傳回子查詢中每個草圖的基數。

```
CREATE TABLE Sales (customer VARCHAR, country VARCHAR, amount BIGINT);
INSERT INTO Sales VALUES ('David Joe', 'Greece', 14.5),  ('David Joe', 'Greece',
 19.95), ('John Doe', 'USA', 29.95), ('John Doe', 'USA', 19.95), ('George Spanos',
 'Greece', 9.95), ('George Spanos', 'Greece', 2.95);
```

以下查詢為會每個國家/地區的客戶產生 HLL 草圖，並擷取基數。這會顯示了來自每個國家/地區的獨特客戶。

```
SELECT hll_cardinality(sketch), country
FROM (SELECT hll_create_sketch(customer) AS sketch, country
        FROM Sales
        GROUP BY country) AS hll_subquery;

hll_cardinality | country
----------------+---------
            1   | USA
            2   | Greece
 ...
```

範例：從子查詢中的組合草圖傳回 HLLSKETCH 類型

下列範例會傳回單一 HLLSKETCH 類型，該類型代表子查詢中個別草圖的組合。草圖會使用 HLL_COMBINE 彙總函數進行組合。

```
SELECT hll_combine(sketch)
FROM (SELECT hll_create_sketch(customers) AS sketch
        FROM Sales
        GROUP BY country) AS hll_subquery
```

```
                                        hll_combine
-----------------------------------------------------------------------------------------------
 {"version":1,"logm":15,"sparse":{"indices":[29808639,35021072,47612452],"values":
[1,1,1]}}
(1 row)
```

範例：從合併多個草圖返回草圖 HyperLogLog

在下列範例中，假設 page-users 資料表會針對使用者在指定網站上造訪的每個頁面儲存預先彙總的草圖。此表格中的每一列都包含一個 HyperLogLog 草圖，代表顯示瀏覽過的頁面的所有使用者 ID。

```
page_users
-- +----------------+-------------+--------------+
-- | _PARTITIONTIME | page        | sketch |
-- +----------------+-------------+--------------+
-- | 2019-07-28     | homepage    | CHAQkAQYA... |
-- | 2019-07-28     | Product A   | CHAQxPnYB... |
-- +----------------+-------------+--------------+
```

下列範例會結合預先彙總的多個草圖並產生單一草圖。此草圖封裝了每個草圖封裝的集合基數。

```
SELECT hll_combine(sketch) as sketch
FROM page_users
```

輸出結果類似如下。

```
-- +----------------------------------------+
-- | sketch |
-- +----------------------------------------+
-- | CHAQ3sGoCxgCIAuCB4iAIBgTIBgqgIAgAwY.... |
-- +----------------------------------------+
```

建立新草圖時，您可以使用 HLL_CARDINALITY 函數來取得集合的不同值，如下所示。

```
SELECT hll_cardinality(sketch)
FROM (
  SELECT
  hll_combine(sketch) as sketch
  FROM page_users
```

```
) AS hll_subquery
```

輸出結果類似如下。

```
-- +-------+
-- | count |
-- +-------+
-- | 54356 |
-- +-------+
```

範例：使用外部資料表在 S3 資料上產生 HyperLogLog 草圖

下列範例快取 HyperLogLog 草圖可避免直接存取 Amazon S3 以進行基數估算。

您可以在定義用來保存 Amazon S3 資料的外部表格中預先彙總和快取 HyperLogLog 草圖。透過這樣做，您可以在不存取基礎資料的情況下擷取基數估計值。

例如，假設您已將一組以 Tab 鍵分隔的文字檔卸載至 Amazon S3。您可以執行下列查詢，以定義 Amazon Redshift 外部結構描述 (名為 `sales`) 中名為 `spectrum` 的外部資料表。此範例的 Amazon S3 儲存貯體位於美國東部 (維吉尼亞北部) AWS 區域。

```
create external table spectrum.sales(
salesid integer,
listid integer,
sellerid smallint,
buyerid smallint,
eventid integer,
dateid integer,
qtysold integer,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t' stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales/';
```

假設您要計算在任意日期購買商品的不同買家。為此，下列範例會為一年中每一天的購買者 ID 產生草圖，並將結果儲存在 Amazon Redshift 資料表 `hll_sales` 中。

```
CREATE TABLE hll_sales AS
SELECT saletime, hll_create_sketch(buyerid) AS sketch
```

```
FROM spectrum.sales
GROUP BY saletime;

SELECT TOP 5 * FROM hll_sales;
```

輸出結果類似如下。

```
-- hll_sales

-- | saletime        | sketch
    |
-- +-----------------
+-------------------------------------------------------------------------+
-- | 7/22/2008 8:30  | {"version":1,"logm":15,"sparse":{"indices":[9281416],"values":
[1]}}
-- | 2/19/2008 0:38  | {"version":1,"logm":15,"sparse":{"indices":[48735497],"values":
[3]}}
-- | 11/5/2008 4:49  | {"version":1,"logm":15,"sparse":{"indices":[27858661],"values":
[1]}}
-- | 10/27/2008 4:08 | {"version":1,"logm":15,"sparse":{"indices":[65295430],"values":
[2]}}
-- | 2/16/2008 9:37  | {"version":1,"logm":15,"sparse":{"indices":[56869618],"values":
[2]}}
-- +----------------
 +-------------------------------------------------------------------------+
```

以下查詢顯示在 2008 年感恩節之後的星期五購買物品的不同買家預計數量。

```
SELECT hll_cardinality(hll_combine(sketch)) as distinct_buyers
FROM hll_sales
WHERE trunc(saletime) = '2008-11-28';
```

輸出結果類似如下。

```
distinct_buyers
---------------
386
```

假設您想要取得特定日期範圍內購買商品的不同使用者數量。例如，可能是從感恩節後的星期五到下一個星期一。為了得到此結果，下列查詢會使用 hll_combine 彙總函數。此函數可避免重複計算在超過所選範圍一天時購買商品的買家。

```
SELECT hll_cardinality(hll_combine(sketch)) as distinct_buyers
FROM hll_sales
WHERE saletime BETWEEN '2008-11-28' AND '2008-12-01';
```

輸出結果類似如下。

```
distinct_buyers
---------------
1166
```

若要保留資hll_sales料表 up-to-date，請在每天結束時執行下列查詢。這樣做會根據今日購買物品的買家帳號產生 HyperLogLog 草圖，並將其加入hll_sales表格中。

```
INSERT INTO hll_sales
SELECT saletime, hll_create_sketch(buyerid)
FROM spectrum.sales
WHERE TRUNC(saletime) = to_char(GETDATE(), 'YYYY-MM-DD')
GROUP BY saletime;
```

跨資料庫查詢資料

透過在 Amazon Redshift 中使用跨資料庫查詢，您可以在 Amazon Redshift 叢集中跨資料庫進行查詢。透過跨資料庫查詢，無論您連線到哪個資料庫，都可以查詢 Amazon Redshift 叢集中任何資料庫的資料。跨資料庫查詢可避免產生資料複本，並簡化您的資料組織，以支援來自相同資料倉儲的多個業務群組。

透過跨資料庫查詢，您可以執行下列操作：

- 查詢 Amazon Redshift 叢集中各個資料庫的資料。

  您不僅可以從連線的資料庫中進行查詢，還可以從您有權存取的任何其他資料庫中讀取資料。

  當您在任何其他未連線的資料庫上查詢資料庫物件時，您只能讀取這些資料庫物件。您可以使用跨資料庫查詢存取 Amazon Redshift 叢集上任何資料庫的資料，而不必連線到該特定資料庫。這樣做可協助您快速輕鬆地查詢和聯結分散在 Amazon Redshift 叢集中多個資料庫的資料。

  您也可以在單一查詢中結合來自多個資料庫的資料集，並使用商業智慧 (BI) 或分析工具來分析資料。您可以使用標準的 Amazon Redshift SQL 命令，繼續為使用者設定精細的表格層級存取控制。如此一來，您可以協助確保使用者只能看到他們有權存取之資料的相關子集。
- 查詢物件。

  您可以使用以三部分標記法表示的完整物件名稱來查詢其他資料庫物件。任何資料庫物件的完整路徑都包含三個元件：資料庫名稱、結構描述和物件名稱。您可以使用完整路徑標記法 (*database_name*.*schema_name*.*object_name*)，從任何其他資料庫存取任何物件。若要存取特定資料欄，請使用 *database_name*.*schema_name*.*object_name*.*column_name*。

  您也可以使用外部結構描述標記法，為另一個資料庫中的結構描述建立別名。此外部結構描述會參照另一個資料庫和結構描述配對 查詢可以使用外部結構描述標記法 (*external_schema_name*.*object_name*)，來存取其他資料庫物件。

  在相同的唯讀查詢中，您可以查詢各種資料庫物件，例如使用者資料表、一般檢視、具體化視觀表，以及來自其他資料庫的近期繫結檢視。
- 管理許可。

  有權存取 Amazon Redshift 叢集中任何資料庫中物件的使用者都可以查詢這些物件。您可以使用 GRANT 命令將權限授予使用者和使用者群組。當使用者不再需要特定資料庫物件的存取權時，您也可以使用 REVOKE 命令撤銷權限。

- 使用中繼資料和 BI 工具。

  您可以建立外部結構描述，參照同一個 Amazon Redshift 叢集內另一個 Amazon Redshift 資料庫中的結構描述。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA 命令。

  建立外部結構描述參照之後，Amazon Redshift 會顯示 SVV_EXTERNAL_TABLES 和 SVV_EXTERNAL_COLUMNS 中其他資料庫結構描述下的資料表，以利工具探索中繼資料。

  若要整合跨資料庫查詢與 BI 工具，您可以使用下列系統檢視。這些檢視可協助您檢視 Amazon Redshift 叢集上已連線資料庫和其他資料庫中物件中繼資料的相關資訊。

  以下是顯示 Amazon Redshift 叢集中所有 Amazon Redshift 物件及所有資料庫的外部物件的系統檢視：

  - SVV_ALL_COLUMNS
  - SVV_ALL_SCHEMAS
  - SVV_ALL_TABLES

  以下是顯示 Amazon Redshift 叢集中所有資料庫的所有 Amazon Redshift 物件的系統檢視：

  - SVV_REDSHIFT_COLUMNS
  - SVV_REDSHIFT_DATABASES
  - SVV_REDSHIFT_FUNCTIONS
  - SVV_REDSHIFT_SCHEMAS
  - SVV_REDSHIFT_TABLES

主題

考量事項

當您使用 Amazon Redshift 中的跨資料庫查詢功能時，請考慮下列事項：

- Amazon Redshift 支援 ra3.4xlarge、ra3.16xlarge 和 ra3.xlplus 節點類型的跨資料庫查詢。

- Amazon Redshift 支援在同一個 Amazon Redshift 叢集中的一個或多個資料庫上聯結來自資料表或檢視的資料。
- Amazon Redshift Serverless 支援與 Amazon Redshift 叢集相同的跨資料庫功能，因此您可以在無伺服器命名空間中的一或多個資料庫中聯結來自資料表或檢視的資料。
- 連線資料庫上交易中的所有查詢都會讀取其他資料庫中狀態相同的資料，也就是交易開始時的資料狀態。此方法有助於跨資料庫提供查詢的交易一致性。Amazon Redshift 支援跨資料庫查詢的交易一致性。
- 若要跨資料庫取得中繼資料，請使用 SVV_ALL* 和 SVV_REDSHIFT* 中繼資料檢視。您不能使用三部分標記法或外部結構描述來查詢 information_schema 和 pg_catalog 下的跨資料庫中繼資料表或檢視。

限制

當您使用 Amazon Redshift 中的跨資料庫查詢功能時，請留意下列限制：

- 當您在任何其他未連線的資料庫上查詢資料庫物件時，您只能讀取這些資料庫物件。
- 若查詢在參照另一個資料庫物件的其他資料庫上建立，則您無法查詢該檢視。
- 您只能在叢集中其他資料庫的物件上建立近期繫結檢視和具體化視觀表。您無法在叢集中其他資料庫的物件上建立一般檢視。
- Amazon Redshift 不支援具有資料欄層級權限的資料表進行跨資料庫查詢。
- Amazon Redshift 不支援 AWS Glue 或聯合資料庫上的查詢目錄物件。若要查詢這些物件，請先建立參照每個資料庫中那些外部資料來源的外部結構描述。
- 不支援在具有交錯排序索引鍵的資料表上執行跨資料庫查詢。

使用跨資料庫查詢的範例

使用下列範例可協助您了解如何設定參考 Amazon Redshift 資料庫的跨資料庫查詢。

若要開始，請在您的 Amazon Redshift 叢集中建立 db1 和 db2 資料庫及使用者 `user1` 和 `user2`。如需詳細資訊，請參閱 CREATE DATABASE 及 CREATE USER。

```
--As user1 on db1
CREATE DATABASE db1;

CREATE DATABASE db2;
```

```
CREATE USER user1 PASSWORD 'Redshift01';

CREATE USER user2 PASSWORD 'Redshift01';
```

作為 db1 上的 user1，建立資料表、授予 user2 存取權限，以及將值插入 table1。如需詳細資訊，請參閱 GRANT 及 INSERT。

```
--As user1 on db1
CREATE TABLE table1 (c1 int, c2 int, c3 int);

GRANT SELECT ON table1 TO user2;

INSERT INTO table1 VALUES (1,2,3),(4,5,6),(7,8,9);
```

作為 db2 上的 user2，使用三部分標記法在 db2 中執行跨資料庫查詢。

```
--As user2 on db2
SELECT * from db1.public.table1 ORDER BY c1;
c1 | c2  | c3
---+-----+----
1  |  2  | 3
4  |  5  | 6
7  |  8  | 9
(3 rows)
```

作為 db2 上的 user2，建立外部結構描述，並使用外部結構描述標記法在 db2 中執行跨資料庫查詢。

```
--As user2 on db2
CREATE EXTERNAL SCHEMA db1_public_sch
FROM REDSHIFT DATABASE 'db1' SCHEMA 'public';

SELECT * FROM db1_public_sch.table1 ORDER BY c1;

c1  | c2 | c3
----+----+----
1   | 2  | 3
4   | 5  | 6
7   | 8  | 9
(3 rows)
```

若要建立不同的檢視並授予這些檢視的許可，作為 db1 上的 user1，請執行以下操作。

```
--As user1 on db1
CREATE VIEW regular_view AS SELECT c1 FROM table1;

GRANT SELECT ON regular_view TO user2;


CREATE MATERIALIZED VIEW mat_view AS SELECT c2 FROM table1;

GRANT SELECT ON mat_view TO user2;


CREATE VIEW late_bind_view AS SELECT c3 FROM public.table1 WITH NO SCHEMA BINDING;

GRANT SELECT ON late_bind_view TO user2;
```

作為 db2 上的 user2，請使用三部分標記法執行下列跨資料庫查詢，以檢視特定檢視。

```
--As user2 on db2
SELECT * FROM db1.public.regular_view;
c1
----
1
4
7
(3 rows)

SELECT * FROM db1.public.mat_view;
c2
----
8
5
2
(3 rows)

SELECT * FROM db1.public.late_bind_view;
c3
----
3
6
9
(3 rows)
```

作為 db2 上的 user2，請使用外部結構描述標記法執行下列跨資料庫查詢，以查詢近期繫結檢視。

```
--As user2 on db2
SELECT * FROM db1_public_sch.late_bind_view;
c3
----
3
6
9
(3 rows)
```

作為 db2 上的 user2，請在單一查詢中使用連線的資料表執行以下命令。

```
--As user2 on db2
CREATE TABLE table1 (a int, b int, c int);

INSERT INTO table1 VALUES (1,2,3), (4,5,6), (7,8,9);

SELECT a AS col_1, (db1.public.table1.c2 + b) AS sum_col2, (db1.public.table1.c3 + c)
 AS sum_col3 FROM db1.public.table1, table1 WHERE db1.public.table1.c1 = a;
col_1 | sum_col2 | sum_col3
------+----------+----------
1     | 4        | 6
4     | 10       | 12
7     | 16       | 18
(3 rows)
```

下列範例會列出叢集上的所有資料庫。

```
select database_name, database_owner, database_type
from svv_redshift_databases
where database_name in ('db1', 'db2');

 database_name | database_owner | database_type
---------------+----------------+---------------
 db1           |            100 | local
 db2           |            100 | local
(2 rows)
```

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 結構描述。

```
select database_name, schema_name, schema_owner, schema_type
```

```
from svv_redshift_schemas
where database_name in ('db1', 'db2');

 database_name |    schema_name     | schema_owner | schema_type
---------------+--------------------+--------------+-------------
 db1           | pg_catalog         |            1 | local
 db1           | public             |            1 | local
 db1           | information_schema |            1 | local
 db2           | pg_catalog         |            1 | local
 db2           | public             |            1 | local
 db2           | information_schema |            1 | local
(6 rows)
```

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 資料表或檢視。

```
select database_name, schema_name, table_name, table_type
from svv_redshift_tables
where database_name in ('db1', 'db2') and schema_name in ('public');

 database_name | schema_name |     table_name      | table_type
---------------+-------------+---------------------+------------
 db1           | public      | late_bind_view      | VIEW
 db1           | public      | mat_view            | VIEW
 db1           | public      | mv_tbl__mat_view__0 | TABLE
 db1           | public      | regular_view        | VIEW
 db1           | public      | table1              | TABLE
 db2           | public      | table2              | TABLE
(6 rows)
```

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 和外部結構描述。

```
select database_name, schema_name, schema_owner, schema_type
from svv_all_schemas where database_name in ('db1', 'db2') ;

 database_name |    schema_name     | schema_owner | schema_type
---------------+--------------------+--------------+-------------
 db1           | pg_catalog         |            1 | local
 db1           | public             |            1 | local
 db1           | information_schema |            1 | local
 db2           | pg_catalog         |            1 | local
 db2           | public             |            1 | local
 db2           | information_schema |            1 | local
 db2           | db1_public_sch     |            1 | external
```

```
(7 rows)
```

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 和外部資料表。

```
select database_name, schema_name, table_name, table_type
from svv_all_tables
where database_name in ('db1', 'db2') and schema_name in ('public');

 database_name | schema_name |     table_name      | table_type
---------------+-------------+---------------------+------------
 db1           | public      | regular_view        | VIEW
 db1           | public      | mv_tbl__mat_view__0 | TABLE
 db1           | public      | mat_view            | VIEW
 db1           | public      | late_bind_view      | VIEW
 db1           | public      | table1              | TABLE
 db2           | public      | table2              | TABLE
(6 rows)
```

搭配查詢編輯器使用跨資料庫查詢

您可以使用跨資料庫查詢存取 Amazon Redshift 叢集上任何資料庫的資料，而不必連線到該特定資料庫。當您在任何其他未連線的資料庫上執行跨資料庫查詢時，您只能讀取這些資料庫物件。

您可以使用以三部分標記法表示的完整物件名稱來查詢其他資料庫物件。任何資料庫物件的完整路徑都包含三個元件：資料庫名稱、結構描述和物件名稱。例如，*database_name.schema_name.object_name*。

搭配查詢編輯器 v2 使用跨資料庫查詢

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 建立叢集以在 Amazon Redshift 查詢編輯器 v2 中使用跨資料庫查詢。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的建立叢集。
3. 使用適當許可啟用查詢編輯器的存取。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的使用查詢編輯器來查詢資料庫。
4. 在導覽選單中，選擇查詢編輯器 v2，然後連線至您叢集的資料庫。

   當您第一次連線到查詢編輯器 v2 時，Amazon Redshift 預設會顯示已連線資料庫的資源。

5. 選擇您有權檢視其資料庫物件的其他資料庫。若要檢視物件，請確定您有適當的許可。選擇資料庫之後，Amazon Redshift 會顯示資料庫中的結構描述清單。

   選取結構描述以查看該結構描述內的資料庫物件清單。

   > **Note**
   >
   > Amazon Redshift 不能直接支援作為 AWS Glue 或聯合資料庫一部分的查詢目錄物件。若要查詢這些物件，請先建立參照每個資料庫中那些外部資料來源的外部結構描述。
   > 具有三部分標記法的 Amazon Redshift 跨資料庫查詢不支援結構描述 `information_schema` 和 `pg_catalog` 下的中繼資料表，因為這些中繼資料檢視專屬於某個資料庫。

6. (選擇性) 篩選您選取之結構描述的資料表或檢視。

在 Amazon Redshift 中共享數據

透過 Amazon Redshift 資料共用，您可以在 Amazon Redshift 叢集、工作群組之間安全地共用即時資料的存取權 AWS 帳戶， AWS 區域 而且無需手動移動或複製資料。由於資料是即時的，所有使用者一旦更新，就可以在 Amazon Redshift 中看到最多 up-to-date 且一致的資訊。

您可以在佈建的叢集、無伺服器工作群組、可用區域和之間共用資料。 AWS 帳戶 AWS 區域您可以在叢集類型之間共用，也可以在佈建的叢集和無伺服器之間共用。

Amazon Redshift 中的多倉庫寫入 (預覽)

您可以在不同的 Amazon Redshift 叢集或 Amazon Redshift 無伺服器工作群組之間共用讀取和寫入的資料庫物件 AWS 帳戶，或從一個叢集到另一個叢集。 AWS 帳戶 您也可以跨區域寫入資料。您可以針對不同的資料表授予 SELECT、INSERT 和 UPDATE 等權限，並針對不同的結構描述授予 USAGE 和 CREATE 權限。一旦確認寫入交易，資料即可供所有倉儲使用。

有關在 PREVIEW_2023 軌道中配置資料共用功能的詳細資訊，請參閱共用資料的寫入存取權限 (預覽)。

> **Note**
>
> ra3.xlplus 叢集目前無法使用透過資料共用的多倉儲寫入功能。若要使用此功能，請建立 ra3.4xl 叢集、ra3.16xl 叢集或 Amazon Redshift 無伺服器工作群組。

Amazon Redshift 中的資料共用概述

透過資料共用，您可以在 Amazon Redshift 叢集之間安全且輕鬆地共用即時資料。

如需有關如何使用開始使用資料共用和管理資料存取的資訊，請參閱。 AWS Management Console管理資料共用工作

Amazon Redshift 的資料共用使用案例

Amazon Redshift 資料共用對於下列使用案例特別有用：

- 支援不同類型的關鍵業務工作負載 — 使用與多個商業智慧 (BI) 或分析叢集共用資料的中央擷取、轉換和載入 (ETL) 叢集。此方法可為個別工作負載提供讀取工作負載隔離和計費。您可以根據工作負載的特定價格和效能需求調整個別工作負載運算的大小和規模。

- 實現跨群組協同合作 — 實現跨團隊和業務群組的無縫協作，以實現更廣泛的分析、資料科學和跨產品影響分析。
- 以服務形式提供資料 — 在整個組織中共用資料即服務。
- 在環境之間共用資料 — 在開發、測試和生產環境之間共用資料。您可以透過分享精細層級不同的資料來提高團隊敏捷性。
- 授權存取 Amazon Redshift 中的資料 — 在 AWS Data Exchange 目錄中列出客戶可以在幾分鐘內找到、訂閱和查詢的 Amazon Redshift 資料集。

資料共用寫入存取使用案例 (預覽)

寫入的資料共享有幾個重要的使用案例：

- 更新生產者的業務來源資料 — 您可以在整個組織中以服務的形式共用資料，但取用者也可以對來源資料執行動作。例如，他們可以傳回 up-to-date 值或確認接收數據。這些只是幾個可能的業務用例。
- 在生產者上插入其他記錄 — 消費者可以將記錄新增至原始來源資料。如果需要，這些可以標記為來自消費者。

如需如何對資料保護執行寫入作業的詳細資訊，請參閱共用資料的寫入存取權限 (預覽)。

在 Amazon Redshift 中共用不同層級的資料

您可以使用 Amazon Redshift 在不同層級上共用資料。這些層次包括資料庫、結構描述、資料表、檢視 (包括一般、近期繫結和具體化視觀表)，以及 SQL 使用者定義函數 (UDF)。您可以為指定資料庫建立多個資料庫。在建立共用的資料庫中，資料共用可以包含多個結構描述的物件。

透過共用資料的這個彈性，您可以獲得精細存取控制。您可以針對需要存取 Amazon Redshift 資料的不同使用者和企業量身打造此控制項。

在 Amazon Redshift 中管理資料一致性

Amazon Redshift 為所有生產者和消費者叢集提供交易一致性，以 up-to-date 及與所有消費者共用資料的一致檢視。

您可以持續更新生產者叢集上的資料。交易中取用者叢集上的所有查詢都會讀取共用資料的相同狀態。Amazon Redshift 不會考慮由生產者叢集上的另一個交易所變更的資料，這些資料是在取用者叢集上的交易開始之後遞交的。在生產者叢集上遞交資料變更之後，取用者叢集上的新交易可以立即查詢更新的資料。

強大的一致性消除了在共用資料期間低保真報告可能包含無效結果的風險。若是財務分析或結果可能用於準備機器學習模型訓練資料集的情況，此要素尤其重要。

在 Amazon Redshift 中使用資料共用時的考量事項

以下是使用 Amazon Redshift 資料共用的考量。如需資料共用限制的資訊，請參閱 資料共用的限制。

- 跨區域資料共用包含額外的跨區域資料傳輸費用。這些資料傳輸費用不適用於同一地區，僅適用於跨區域。如需詳細資訊，請參閱 管理跨區域資料共用的成本控制。
- 身為資料共用使用者，您只能繼續連線到本機叢集資料庫。您無法連線到從資料共用中建立的資料庫，但可以從這些資料庫中讀取。
- 取用者需支付查詢生產者資料所需的所有運算和跨區域資料傳輸費用。生產者須支付其佈建叢集或無伺服器命名空間中資料的基礎儲存費用。
- 查詢共用資料的效能取決於取用者叢集的運算容量。

管理叢集加密

若要跨越共用資料 AWS 帳戶，生產者和消費者叢集都必須加密。

在 Amazon Redshift 中，您可以為您的叢集開啟資料庫加密，以協助保護靜態資料。開啟叢集的加密時，叢集和其快照的資料區塊和系統中繼資料會加密。您可以在啟動叢集時開啟加密，或修改未加密的叢集來使用 AWS Key Management Service (AWS KMS) 加密。如需 Amazon Redshift 資料庫加密的相關資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 資料庫加密。

為了保護傳輸中的資料，傳輸過程中的所有資料都會透過生產者叢集的加密結構描述加密。取用者叢集會在載入資料時採用此加密結構描述。取用者叢集接著會以一般加密叢集的形式運作。生產者和取用者之間的通訊也會使用共用金鑰結構描述進行加密。如需傳輸中加密的相關資訊，請參閱傳輸中加密。

資料共用的限制

以下是在 Amazon Redshift 中使用資料共用時的限制：

- 所有佈建的 ra3 叢集類型 (ra3.16 x 大型、ra3.4 x 大型和 ra3.xlplus) 和 Amazon Redshift 無伺服器都支援資料共用。它不支援其他叢集類型。
- 對於跨帳戶和跨區域資料共用，生產者和取用者叢集及無伺服器命名空間都必須加密。這是出於安全目的。但是，他們不需要共用相同的加密金鑰。
- 您只能透過資料共用來共用 SQL UDF。不支援 Python 和 Lambda UDF。
- 如果生產者資料庫具有特定的定序，請為取用者資料庫使用相同的定序設定。

- Amazon Redshift 不支援將外部結構描述、資料表或外部資料表上的近期繫結檢視新增至資料共用。
- Amazon Redshift 不支援生產者叢集上的巢狀 SQL 使用者定義函式。
- Amazon Redshift 不支援使用交錯排序索引鍵的共用資料表，以及參考具有交錯排序索引鍵之資料表的檢視。
- 取用者不能將資料共用物件新增到另一個資料共用。此外，取用者無法將參考資料共用物件的檢視新增到另一個資料共用中。
- Amazon Redshift 不支援存取同時在準備和執行存取之間發生 DDL 的資料共用物件。
- Amazon Redshift 不支援透過資料共用來共用預存程序。
- Amazon Redshift 不支援共用中繼資料系統檢視和系統資料表。

提供資料共用的區域

下表列出資料共用功能的可用性。

區域	同一地區資料共用	跨區域資料共用	AWS Lake Formation 控管的資料共用
美國東部 (維吉尼亞北部) (us-east-1)	是	是	是
美國東部 (俄亥俄) (us-east-2)	是	是	是
美國西部 (加利佛尼亞北部) (us-west-1)	是	是	是
美國西部 (奧勒岡) (us-west-2)	是	是	是
亞太區域 (孟買) (ap-south-1)	是	是	是
亞太區域 (海德拉巴) (ap-south-2)	是	否	否
亞太區域 (東京) (ap-northeast-1)	是	是	是

區域	同一地區資料共用	跨區域資料共用	AWS Lake Formation 控管的資料共用
亞太區域 (新加坡) (ap-southeast-1)	是	是	是
亞太區域 (雪梨) (ap-southeast-2)	是	是	是
亞太區域（雅加達）；（ap-southeast-3）	是	否	否
亞太區域 (墨爾本) (ap-southeast-4)	是	否	否
亞太區域 (首爾) (ap-northeast-2)	是	是	是
亞太區域 (大阪) (ap-northeast-3)	是	否	否
非洲 (開普敦) (af-south-1)	是	是	否
加拿大西部 (卡加利) (ca-west-1)	是	否	否
加拿大 (中部) (ca-central-1)	是	是	是
歐洲 (法蘭克福) (eu-central-1)	是	是	是
歐洲 (蘇黎世) (eu-central-2)	是	否	否
歐洲 (愛爾蘭) (eu-west-1)	是	是	是

區域	同一地區資料共用	跨區域資料共用	AWS Lake Formation 控管的資料共用
歐洲 (倫敦) (eu-west-2)	是	是	是
歐洲 (巴黎) (eu-west-3)	是	是	是
歐洲 (米蘭) (eu-south-1)	是	否	否
歐洲 (西班牙) (eu-south-2)	是	否	否
歐洲 (斯德哥爾摩) (eu-north-1)	是	是	是
中東 (阿拉伯聯合大公國) (me-central-1)	是	否	否
中東 (巴林) (me-south-1)	是	否	否
以色列 (特拉維夫) (il-central-1)	是	否	否
南美洲 (聖保羅) (sa-east-1)	是	是	是
AWS GovCloud (美國東部) (美國往東 1)	是	否	是
AWS GovCloud (美國西部) (美國-往西 -1)	是	否	是

多倉儲寫入資料共用的區域可用性

在 PREVIEW_2023 軌道中，數據共享具有寫操作和更精細的共享功能的能力。如需如何設定這些項目的詳細資訊，請參閱共用資料的寫入權限 (預覽)。如需有關提供預覽功能之區域的詳細資訊，請參閱提供資料共用的區域 (預覽)。

什麼是資料共用？

資料共用是在 Amazon Redshift 中共用資料的單位。使用數據庫共享相同或不同 AWS 帳戶 的數據。AWS 帳戶此外，也可以在不同的 Amazon Redshift 叢集之間共用資料以供讀取用。

每個資料共用都與 Amazon Redshift 叢集中的特定資料庫相關聯。

生產者叢集管理員可以建立資料共用，並新增資料共用物件以與其他叢集共用資料，此動作稱為輸出共用。取用者叢集管理員可以從其他叢集接收資料共用，此動作稱為輸入共用。有關生產者和取用者的詳細資訊，請參閱 資料共用生產者和取用者。

Datashare 物件是叢集上特定資料庫的物件，生產者叢集管理員可以將其新增至資料共用，以便與資料取用者共用。對於資料取用者而言，資料共用物件是唯讀的。資料共用物件的範例包括資料表、檢視和使用者定義函數。建立資料共用或編輯資料共用時，您可以隨時將資料共用物件新增至資料共用。

調整叢集大小或暫停生產者叢集時，資料共用會繼續運作。

資料共用有不同的類型。

主題

- 標準資料共用
- AWS Data Exchange 資料庫
- AWS Lake Formation管理的資料共用
- 資料共用生產者和取用者

標準資料共用

使用標準資料存檔，您可以在佈建的叢集、無伺服器工作群組、可用區域和. AWS 帳戶 AWS 區域您可以在叢集類型之間共用，也可以在佈建的叢集和 Amazon Redshift Serverless 之間共用。

若要共用資料，請注意下列佈建的叢集、無伺服器命名空間和 AWS 帳戶 識別碼：

- 佈建的叢集命名空間是識別 Amazon Redshift 佈建叢集的識別碼。命名空間全域唯一識別碼 (GUID) 會在建立佈建的叢集時自動建立，並附加至叢集。Amazon Resource Name (ARN) 命名空間會使用

arn:{partition}:redshift:{region}:{account-id}:namespace:{namespace-guid} 格式。您可以在 Amazon Redshift 主控台的叢集詳細資料頁面上查看已佈建叢集的命名空間。

在資料共用工作流程中，命名空間 GUID 值和叢集命名空間 ARN 會用於與 AWS 帳戶中的叢集共用資料。您也可以使用 `current_namespace` 函數尋找目前叢集的命名空間。

- 無伺服器命名空間是識別 Amazon Redshift Serverless 的識別碼。命名空間全域唯一識別碼 (GUID) 會在建立 Amazon Redshift Serverless 時自動建立，並附加至執行個體。無伺服器命名空間 ARN 使用 arn:{partition}:redshift-serverless:{region}:{account-id}:namespace/{namespace-guid} 格式。
- AWS 帳戶 可以是數據存儲器的消費者，並且每個用 12 位數 ID 表示。 AWS 帳戶

對於標準共用，請考量到下列內容：

- 刪除生產者叢集時，Amazon Redshift 會刪除生產者叢集所建立的資料共用。備份和還原生產者叢集時，建立的資料共用仍會存在還原的叢集上。但是，授予其他叢集的資料共用權限在還原的叢集上不再有效。警將資料共用的使用許可重新授予所需的取用者叢集。取用者叢集上的取用者資料庫會指向建立快照集之原始叢集的資料共用。若要從還原的叢集查詢共用資料，取用者叢集管理員會建立不同的資料庫。或者，管理員可以捨棄並重新建立現有的使用者資料庫，以使用新還原叢集中的資料共用。
- 刪除取用者叢集並從快照中將其還原時，先前與此叢集共用的存取權將不再有效，也不再可見。如果還原的取用者叢集上仍需要資料共用的存取權，則生產者叢集管理員必須再次將資料共用的使用權授予已還原的取用者叢集。取用者叢集管理員必須捨棄從非作用中資料庫建立的任何過時取用者資料庫。然後，管理員必須在生產者重新授予許可之後，從資料共用中重新建立取用者資料庫。由於從原始叢集還原的叢集上的叢集命名空間 GUID 不同，因此當取用者或生產者叢集從備份還原時，請重新授予資料共用許可。

AWS Data Exchange 資料庫

資 AWS Data Exchange 料識別是透過共用資料的授權單位。 AWS Data Exchange AWS 管理與訂閱和使用 Amazon Redshift 資料共用相關聯的所有帳單 AWS Data Exchange 和付款。核准的資料提供者可以將資料 AWS Data Exchange 庫新增至產品。 AWS Data Exchange 當客戶訂閱具有 AWS Data Exchange 資料存取權的產品時，他們可以存取產品中的資料庫。

AWS Data Exchange 對於 Amazon Redshift 可以方便地通過授權訪問您的 Amazon Redshift 數據。AWS Data Exchange當客戶訂閱含有資 AWS Data Exchange 料庫的產品時， AWS Data Exchange 會自動將客戶新增為產品隨附的所有 AWS Data Exchange 資料庫上的資料消費者。系統會自動生成發票，並集中收集付款並自動支付。 AWS Marketplace Entitlement Service

提供者可以在 Amazon Redshift 中以細微層級授權資料，例如結構描述、資料表、檢視和使用者定義的函數。您可以在多 AWS Data Exchange 個產品中使用相同的 AWS Data Exchange 資料清單。任何新增至 AWS Data Exchange 資料清單的物件都可供取用者使用。生產者可以使用 Amazon Redshift API 操作、SQL 命令和 Amazon Redshift 主控台，檢視由 AWS Data Exchange 其代表其管理的所有 AWS Data Exchange 資料庫。訂閱產品 AWS Data Exchange 資料庫的客戶對資料庫中的物件具有唯讀存取權。

想要使用第三方生產者資料的客戶可以瀏覽 AWS Data Exchange 目錄，探索並訂閱 Amazon Redshift 中的資料集。 AWS Data Exchange 訂閱啟用後，他們可以從叢集中的資料庫建立資料庫，並在 Amazon Redshift 中查詢資料。

AWS Data Exchange 數據庫如何工作

以製作 AWS Data Exchange 者管理員身分管理資料庫

如果您是資料生產者 (也稱為提供者 AWS Data Exchange)，則可以建立連線到 Amazon Redshift 資 AWS Data Exchange 料庫的資料庫。若要在上新增 AWS Data Exchange 資料庫至產品 AWS Data Exchange，您必須是已註冊的提供者。 AWS Data Exchange

如需如何開始使用資 AWS Data Exchange 料存取的詳細資訊，請參閱。共享授權的 Amazon Redshift 數據 AWS Data Exchange

使用 AWS Data Exchange 資料庫作為具有作用中訂閱的消費者 AWS Data Exchange

如果您是具有有效 AWS Data Exchange 訂閱的消費者 (也稱為訂閱者 AWS Data Exchange)，則可以在 AWS Data Exchange 主控台上瀏覽 AWS Data Exchange 目錄以探索包含 AWS Data Exchange 資料存取的產品。

訂閱包含資 AWS Data Exchange 料庫的產品後，請從叢集中的資料清單建立資料庫。然後，您可以直接在 Amazon Redshift 中查詢資料，而無需擷取、轉換和載入資料。

如需如何開始使用資 AWS Data Exchange 料存取的詳細資訊，請參閱。共享授權的 Amazon Redshift 數據 AWS Data Exchange

對於 AWS Data Exchange 共用，請考量到下列內容：

- 刪除生產者叢集時，Amazon Redshift 會刪除生產者叢集所建立的資料共用。備份和還原生產者叢集時，建立的資料共用仍會存在還原的叢集上。若要讓資料訂閱者能夠繼續存取資料，請再次建立資 AWS Data Exchange 料庫並將其發佈至產品的資料集。取用者叢集上的取用者資料庫會指向建立快照集之原始叢集的資料共用。若要查詢還原叢集中的共用資料，用戶叢集管理員會建立不同的資料

庫，或卸除並重新建立現有的用戶資料庫，以使用新還原叢集中新建立 AWS Data Exchange 的資料清單。

- 刪除取用者叢集並從快照中將其還原時，先前與此叢集共用的存取權仍將有效且可見。取用者叢集管理員必須捨棄從非作用中資料共用建立的任何過時取用者資料庫，並在生產者重新授予許可之後，從資料共用重新建立取用者資料庫。由於從原始叢集還原的叢集上的叢集命名空間 GUID 不同，因此當生產者叢集從備份還原時，請重新授予資料共用許可。
- 如果您有任何 AWS Data Exchange 資料存取，建議您不要刪除叢集。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。

使用 Amazon Redshift 時 AWS Data Exchange 的注意事項

使用 AWS Data Exchange Amazon Redshift 時，請考慮以下事項：

- 生產者和取用者都必須使用 RA3 執行個體類型才能使用 Amazon Redshift 資料共用。生產者必須搭配最新的 Amazon Redshift 叢集版本使用 RA3 執行個體類型。
- 生產者和取用者叢集都必須加密。
- 您必須註冊為 AWS Data Exchange 供應商，才能在其上刊登產品 AWS Data Exchange，包括包含 AWS Data Exchange 資料庫的產品。如需詳細資訊，請參閱開始成為提供者。
- 您不需要是註冊的 AWS Data Exchange 供應商，就能透過 AWS Data Exchange尋找、訂閱和查詢 Amazon Redshift 資料。
- 若要控制對資料的存取，請在開啟可公開存取設定的情況下建立資料 AWS Data Exchange 庫。要更改數 AWS Data Exchange 據標題以關閉可公開訪問的設置，請將會話變量設置為允許更改數據保護設置可公開訪問 FALSE。如需詳細資訊，請參閱 ALTER DATASHARE 使用須知。
- 生產者無法手動從 AWS Data Exchange 資料封中新增或移除取用者，因為資料存取權是根據對包含資料清單的產品擁有有效訂閱而授予的 AWS Data Exchange 。 AWS Data Exchange
- 生產者無法檢視取用者執行的 SQL 查詢。他們只能透過只有生產者可以存取的 Amazon Redshift 資料表，檢視中繼資料，例如查詢數量或取用者查詢的物件。如需詳細資訊，請參閱 在 Amazon Redshift 中監視和稽核資料共用。
- 建議您將資料共用設為可公開存取。如果不這樣做，使用可公開存取 AWS Data Exchange 的消費者叢集上的訂閱者將無法使用您的資料清單。
- 我們建議您不要刪除使用 DROP AWS Data Exchange DATASHARE 陳述式共 AWS 帳戶 用給其他人的資料清單。如果 AWS 帳戶 這樣做，可以訪問數據保護的將失去訪問權限。此動作不可復原。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。如果您要刪除 AWS Data Exchange 資料清單，請參閱。DROP DATASHARE 使用須知
- 對於跨區域資料共用，您可以建立資 AWS Data Exchange 料庫以共用授權資料。

- 使用來自不同區域的資料時，取用者須支付從生產者區域到取用者區域的跨區域資料傳輸費用。

AWS Lake Formation管理的資料共用

您可以使用 AWS Lake Formation，集中定義和強制執行 Amazon Redshift 資料庫的資料庫、表格、欄和列層級存取權限，並限制使用者對資料識別中物件的存取權限。透過 Lake Formation 共用資料，您可以在 Lake Formation 中定義許可，並將這些許可套用至任何資料共用及其物件。例如，如果您有一份包含員工資訊的資料表，則可以使用 Lake Formation 的資料欄層級篩選，防止不在人力資源部門工作的員工看到個人身分識別資訊 (PII)，例如社會安全號碼。如需資料篩選的相關資訊，請參閱《AWS Lake Formation 開發人員指南》中的 Lake Formation 中的資料篩選與儲存格層級安全性。

您也可以在 Lake Formation 中使用標籤來設定 Lake Formation 資源的許可。如需詳細資訊，請參閱 Lake Formation 的標籤式存取控制。

Amazon Redshift 目前在相同帳戶內或跨帳戶共用時，可支援透過 Lake Formation 進行資料共用。目前不支援跨區域共用。

以下是如何使用 Lake Formation 控制資料共用許可的概觀：

1. 在 Amazon Redshift 中，生產者叢集或工作群組管理員會在生產者叢集或工作群組上建立資料共用，並將使用權授予 Lake Formation 帳戶。
2. 生產者叢集或工作群組管理員可授權 Lake Formation 帳戶存取資料共用。
3. Lake Formation 管理員可探索並註冊資料共用。他們還必須發現他們可以訪問的 AWS Glue ARN，並將數據庫與 ARN 相關聯。 AWS Glue Data Catalog 如果您正在 AWS CLI 使用，則可以使用 Redshift CLI 操作和. `describe-data-shares` `associate-data-share-consumer` 若要註冊資料共用，請使用 Lake Formation CLI 操作 `register-resource`。
4. Lake Formation 管理員會在中建立聯合資料庫 AWS Glue Data Catalog，並設定 Lake Formation 權限，以控制使用者對資料清單內物件的存取。如需中的聯合資料庫的詳細資訊 AWS Glue，請參閱管理 Amazon Redshift 資料存取中資料的許可。
5. Lake Formation 管理員發現他們有權訪問的數據 AWS Glue 庫，並將數據存儲器與 ARN 相關聯。 AWS Glue Data Catalog
6. Redshift 管理員會探索他們有權存取的 AWS Glue 資料庫 ARN，使用資料庫 ARN 在 Amazon Redshift 取用者叢集中建立外部 AWS Glue 資料庫，並授與使用 IAM 登入資料驗證的資料庫使用者以開始查詢 Amazon Redshift 資料庫。
7. 資料庫使用者可以使用檢視 SVV_EXTERNAL_TABLES 和 SVV_EXTERNAL_COLUMNS 來尋找資料庫內的所有資料表或資料行，他們可以查詢 AWS Glue 資料庫的資料表。 AWS Glue

8. 當生產者叢集或工作群組管理員決定不再與取用者叢集共用資料時，生產者叢集管理員可以撤銷使用權、取消授權或從 Redshift 刪除資料共用。Lake Formation 中的相關許可和物件不會自動刪除。

若要取得有關 AWS Lake Formation 以生產者叢集或工作群組管理員身分共用資料保護的更多資訊，請參閱〈〉。以生產者身分使用 Lake Formation 管理的資料共用若要取用生產者叢集或工作群組中的共用資料，請參閱 以取用者身分使用 Lake Formation 管理的資料共用。

AWS Lake Formation 與 Amazon Redshift 搭配使用時的注意事項和限制

以下是透過 Lake Formation 共用 Amazon Redshift 資料的考量和限制。如需有關資料共用考量和限制的資訊，請參閱在 Amazon Redshift 中使用資料共用時的考量事項。如需 Lake Formation 限制的相關資訊，請參閱有關在 Lake Formation 中使用 Amazon Redshift 資料庫的注意事項。

- 目前不支援跨區域與 Lake Formation 共用資料共用。
- 如果已針對共用關係上的使用者定義資料欄層級篩選，則執行 `SELECT *` 操作只會傳回使用者有權存取的資料欄。
- 不支援 Lake Formation 的儲存格層級篩選。
- 如果您已建立檢視及其資料表並將其共用至 Lake Formation，則可以設定篩選來管理資料表的存取權，Amazon Redshift 會在取用者叢集使用者存取共用物件時強制執行 Lake Formation 定義的政策。當使用者存取與 Lake Formation 共用的檢視時，Redshift 只會強制執行檢視 (而非檢視中資料表) 上定義的 Lake Formation 政策。但是，當使用者直接存取資料表時，Redshift 會強制執行資料表上已定義的 Lake Formation 政策。
- 如果資料表已設定 Lake Formation 篩選，則您無法根據共用資料表建立取用者的具體化視觀表。
- Lake Formation 管理員必須具有 資料湖管理員許可及接受資料共用所需的許可。
- 生產者取用者叢集必須是具有最新 Amazon Redshift 叢集版本的 RA3 叢集，或是無伺服器工作群組，才能透過 Lake Formation 共用資料共用。
- 生產者和取用者叢集都必須加密。
- 將資料共用與 Lake Formation 共用時，在生產者叢集或工作群組中實作的 Redshift 資料列層級和資料欄層級存取控制政策會遭到忽略。Lake Formation 管理員必須在 Lake Formation 中設定這些政策。生產者叢集或工作群組管理員可以使用 ALTER TABLE 命令來關閉資料表的 RLS。
- 透過 Lake Formation 共用資料共用僅適用於可同時存取 Redshift 和 Lake Formation 的使用者。

資料共用生產者和取用者

資料生產者 (也稱為資料共用生產者 (data sharing producer 或 datashare producer)) 是您要從中共用資料的叢集。生產者叢集管理員和資料庫擁有者可以使用 CREATE DATASHARE 命令來建立資料共用。您可以從要產生器叢集與用戶叢集共用的資料庫，新增綱要、資料表、檢視表和 SQL 使用者定義函數 (UDF) 等物件。

資料 AWS Data Exchange 庫的資料生產者 (也稱為提供者 AWS Data Exchange) 可透過授權資料。AWS Data Exchange核准的提供者可以將 AWS Data Exchange 資料庫新增至產品。 AWS Data Exchange

當客戶訂閱含有資 AWS Data Exchange 料庫的產品時， AWS Data Exchange 會自動將客戶新增為產品隨附的所有 AWS Data Exchange 資料庫上的資料消費者。 AWS Data Exchange 也會在訂閱結束時將所有客戶從 AWS Data Exchange 資料庫中移除。 AWS Data Exchange 還可以自動管理帶有 AWS Data Exchange 數據庫的付費產品的帳單，發票開立，付款收款和付款分配。如需詳細資訊，請參閱 AWS Data Exchange 資料庫。若要註冊為 AWS Data Exchange 資料提供者，請參閱開始成為提供者。

資料取用者 (也稱為資料共用取用者 (data sharing consumer 或 datashare consumer)) 是從生產者叢集接收資料共用的叢集。

共用資料的 Amazon Redshift 叢集可以位於相同、不同 AWS 帳戶 或不同 AWS 區域，因此您可以跨組織共用資料並與其他方協同合作。取用者叢集管理員會接收其獲得使用權的資料共用，並檢閱每個資料共用的內容。若要使用共用的資料，取用者叢集管理員會從資料共用中建立 Amazon Redshift 資料庫。然後，管理員會將資料庫的許可指派給取用者叢集中的使用者和角色。授予許可之後，使用者和角色可以將共用物件列為標準中繼資料查詢的一部分，以及取用者叢集上的本機資料。他們可以立即開始查詢。

如果您是擁有有效訂 AWS Data Exchange 閱的消費者 (也稱為訂閱者 AWS Data Exchange)，您可以在 Amazon Redshift 中尋找、訂閱和查詢精細的 up-to-date 資料，而不需要擷取、轉換和載入資料。如需詳細資訊，請參閱 AWS Data Exchange 資料庫。

資料共用在 Amazon Redshift 中的運作方式

管理不同狀態的資料共用

使用跨帳戶資料共用時，會有不同的資料共用狀態，需要您執行動作。您的資料共用狀態包含作用中、需要採取動作或非作用中。

以下說明每個資料共用狀態及其必要動作：

- 當生產者叢集管理員建立資料共用時，生產者叢集上的資料共用狀態為待定授權。生產者叢集管理員可以授權資料取用者存取資料共用。取用者叢集管理員沒有任何動作。
- 當生產者叢集管理員授權資料共用時，生產者叢集上的資料共用狀態會變成已授權。生產者叢集管理員沒有任何動作。當資料共用的資料取用者至少有一個關聯時，資料共用狀態會從已授權變更為作用中。

  接著，取用者叢集上資料共用的共用狀態會變成可用 (需要在 Amazon Redshift 主控台上採取動作)。取用者叢集管理員可以將資料共用與資料取用者建立關聯，或拒絕資料共用。取用者叢集管理員也可以使用 AWS CLI 命令 `describeDatashareforConsumer` 來檢視資料共用的狀態。或者，管理員可以使用 CLI 命令 `describeDatashare` 並提供資料共用 Amazon Resource Name (ARN) 來檢視資料共用的狀態。
- 當取用者叢集管理員將資料共用與資料取用者建立關聯時，生產者叢集上的資料共用狀態會變成作用中。當資料共用的資料取用者至少有一個關聯時，資料共用狀態會從已授權變更為作用中。生產者叢集管理員不需要執行任何動作。

  取用者叢集上的資料共用狀態會變成作用中。取用者叢集管理員不需要執行任何動作。
- 當取用者叢集管理員從資料共用中移除取用者關聯時，資料共用狀態會變成作用中或已授權。當資料共用與另一個資料取用者之間至少存在一個關聯時，它會變成作用中。當生產者叢集上的資料共用沒有任何取用者關聯時，它會變成已授權。生產者叢集管理員沒有任何動作。

  如果移除所有關聯，則資料共用狀態會在取用者叢集上變成需要採取動作。當資料共用可供取用者使用時，取用者叢集管理員可以將資料共用與資料取用者重新建立關聯。
- 當取用者叢集管理員拒絕資料共用時，生產者叢集上的資料共用狀態會在取用者叢集上變成需要採取動作和已拒絕。生產者叢集管理員可以重新授權資料共用。取用者叢集管理員沒有任何動作。
- 當生產者叢集管理員從資料共用中移除授權時，生產者叢集上的資料共用狀態會變成需要採取動作。如有必要，生產者叢集管理員可以選擇重新授權資料共用。取用者叢集管理員不需要執行任何動作。

共用資料共用

只有在不同的 Amazon Redshift 佈建叢集或無伺服器工作群組之間共用資料時，才需要資料共用。在同一個叢集中，只要您擁有其他資料庫中物件的必要權限，就可以使用簡單的三部分標記法 `database.schema.table` 來查詢另一個資料庫。

在 Amazon Redshift 中管理資料共用的許可

身為生產者叢集管理員，您可以保留所共用資料集的控制權。您可以將新物件新增至資料共用，或從資料共用中移除物件。您也可以授與或撤銷取用者叢集、 AWS 帳戶或區域的整體資料存取權。

AWS 當許可被撤銷時，取用者叢集會立即失去共用物件的存取權，並無法在 SVV_DATASHARES 的 INBOUND 資料共用清單中看到共用物件。

下列範例會建立資料清單salesshare、新增結構描述public，並將資料表public.tickit_sales_redshift新增至。salesshare它也會授與指定叢集命名空間的使用權限。salesshare

```
CREATE DATASHARE salesshare;

ALTER DATASHARE salesshare ADD SCHEMA public;

ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;

GRANT USAGE ON DATASHARE salesshare TO NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

對於 CREATE DATASHARE，超級使用者和資料庫擁有者可以建立資料庫。如需詳細資訊，請參閱 CREATE DATASHARE。對於 ALTER DATASHARE，在要新增或移除的資料共用物件上有所需許可的資料共用擁有者可以修改資料共用。如需相關資訊，請參閱ALTER DATASHARE。

身為生產者管理員，當您捨棄資料共用時，資料共用不會再列在取用者叢集上。使用者叢集上從捨棄的資料共用中建立的資料庫和結構描述參考會繼續存在，但其中沒有任何物件。取用者叢集管理員必須手動刪除這些資料庫。

在取用者方面，取用者叢集管理員可以透過從資料共用建立資料庫來決定哪些使用者和角色應該可以存取共用資料。根據您在建立資料庫時所選擇的選項，您可以按照下列方式控制對資料庫的存取。如需從資料共用建立資料庫的詳細資訊，請參閱 CREATE DATABASE。

建立沒有 WITH PERMISSIONS 子句的資料庫

管理員可以在資料庫或結構描述層級上控制存取權。若要在結構描述層級上控制存取權，管理員必須從在資料共用中建立的 Amazon Redshift 資料庫建立外部結構描述。

下列範例會授予在資料庫層級和結構描述層級上存取共用資料表的許可。

```
GRANT USAGE ON DATABASE sales_db TO Bob;

CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE sales_db SCHEMA 'public';

GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

若要進一步限制存取權，您可以在共用物件頂端建立檢視，僅公開必要的資料。然後，您可以使用這些視觀表將存取權提供給使用者和角色。

一旦授予使用者資料庫或結構描述的存取權後，他們就可以存取該資料庫或結構描述中的所有共用物件。

使用 WITH PERMISSIONS 子句建立資料庫

授予資料庫或結構描述的使用權之後，系統管理員可以使用與本機資料庫或結構描述相同的權限授予程序，進一步控制存取權。如果沒有個別物件權限，使用者就無法存取資料附屬資料庫或結構描述中的任何物件，即使在被授予 USAGE 權限之後也一樣。

下列範例會授予在資料庫層級上存取共用資料表的許可。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
GRANT USAGE FOR SCHEMAS IN DATABASE sales_db TO Bob;
GRANT SELECT ON sales_db.public.tickit_sales_redshift TO Bob;
```

授予資料庫或結構描述的存取權之後，使用者仍然需要獲得資料庫或結構描述中任何您要讓他們存取之物件的相關權限。

使用「使用權限」進行精細共享 (預覽

讓叢集或 Serverless 工作群組查詢資料共用

此步驟假設資料共用源自您帳戶中的其他叢集或 Amazon Redshift Serverless 命名空間，或者來自另一個帳戶，且已與您正在使用的命名空間相關聯。

1. 取用者資料庫管理員可以從資料共用建立資料庫。

   ```
   CREATE DATABASE my_ds_db [WITH PERMISSIONS] FROM DATASHARE my_datashare OF
    NAMESPACE 'abc123def';
   ```

   如果您建立具有權限的資料庫，您可以將資料共用物件的細微權限授予不同的使用者和角色。如果沒有這個功能，所有被授予資料共用資料庫 USAGE 權限的使用者和角色，都會被授予資料共用資料庫內所有物件的所有權限。
2. 以下說明如何將權限授予 Redshift 資料庫使用者或角色。您必須連線到本機資料庫，才能執行這些陳述式。如果您在執行授予陳述式之前，在資料共用資料庫上執行 USE 命令，則無法執行這些陳述式。

```
GRANT USAGE ON DATABASE my_ds_db TO ROLE data_eng;
GRANT CREATE, USAGE ON SCHEMA my_ds_db.my_shared_schema TO ROLE data_eng;
GRANT ALL ON ALL TABLES IN SCHEMA my_ds_db.my_shared_schema TO ROLE data_eng;

GRANT USAGE ON DATABASE my_ds_db TO bi_user;
GRANT USAGE ON SCHEMA my_ds_db.my_shared_schema TO bi_user;
GRANT SELECT ON my_ds_db.my_shared_schema.table1 TO bi_user;
```

在 Amazon Redshift 資料共用中使用檢視

生產者叢集可以共用一般、近期繫結和具體化視觀表。共用一般檢視或近期繫結檢視時，您不需要共用基底資料表。下表顯示資料共用如何支援檢視。

檢視表名稱	可以將此檢視新增到資料共用嗎？	取用者可以在跨叢集的資料共用物件上建立此檢視嗎？
一般檢視	是	否
近期繫結檢視	是	是
具體化視觀表	是	是，但僅限完整的重新整理

下面的查詢顯示資料共用支援的一般檢視輸出。如需一般檢視定義的資訊，請參閱 CREATE VIEW。

```
SELECT * FROM tickit_db.public.myevent_regular_vw
ORDER BY eventid LIMIT 5;

   eventid  |  eventname
  ----------+-------------
     3835   | LeAnn Rimes
     3967   | LeAnn Rimes
     4856   | LeAnn Rimes
     4948   | LeAnn Rimes
     5131   | LeAnn Rimes
```

下面的查詢顯示資料共用支援的近期繫結檢視輸出。如需近期繫結檢視的詳細資訊，請參閱 CREATE VIEW。

```
SELECT * FROM tickit_db.public.event_lbv
ORDER BY eventid LIMIT 5;

 eventid | venueid | catid | dateid |          eventname           |      starttime
 --------+---------+-------+--------+------------------------------
+---------------------
     1   |   305   |   8   |  1851  |        Gotterdammerung       | 2008-01-25
 14:30:00
     2   |   306   |   8   |  2114  |           Boris Godunov      | 2008-10-15
 20:00:00
     3   |   302   |   8   |  1935  |              Salome          | 2008-04-19
 14:30:00
     4   |   309   |   8   |  2090  |  La Cenerentola (Cinderella) | 2008-09-21
 14:30:00
     5   |   302   |   8   |  1982  |          Il Trovatore        | 2008-06-05
 19:00:00
```

下面的查詢顯示資料共用支援的具體化視觀表輸出。如需具體化視觀表的詳細資訊，請參閱 CREATE MATERIALIZED VIEW。

```
SELECT * FROM tickit_db.public.tickets_mv;

  catgroup | qtysold
 ----------+---------
  Concerts | 195444
    Shows  | 149905
```

您可以維護生產者叢集中所有租用戶的通用資料表。您也可以與取用者叢集共用依維度資料欄 (例如 tenant_id (account_id 或 namespace_id)) 篩選的資料子集。若要這麼做，您可以在基底資料表上定義檢視，並在這些 ID 欄上使用篩選，例如 current_aws_account = tenant_id。在取用者方面，當您查詢檢視時，您只會看到符合您帳戶資格的資料列。若要這麼做，您可以使用 Amazon Redshift 內容函數 current_aws_account 和 current_namespace。

下列查詢會傳回目前 Amazon Redshift 叢集所在的帳戶 ID。如果您已連線到 Amazon Redshift，則可以執行此查詢。

```
select current_user, current_aws_account;
```

```
current_user | current_aws_account
-------------+--------------------
dwuser       |    111111111111
(1row)
```

下列查詢會傳回目前 Amazon Redshift 叢集的命名空間。如果您已連線到資料庫，則可以執行此查詢。

```
select current_user, current_namespace;

current_user | current_namespace
-------------+--------------------------------------
dwuser       | 86b5169f-01dc-4a6f-9fbb-e2e24359e9a8
(1 row)
```

資料命名中具體化視觀表的累加式重新整理

Amazon Redshift 支援在共用基礎資料表時，針對消費者資料保護中的具體化視圖進行累加式重新整理。累加式重新整理是一項操作，Amazon Redshift 會識別基礎資料表或上次重新整理後發生的資料表中的變更，並僅更新具體化視觀表中的對應記錄。如需此行為的相關資訊，請參閱建立具體化視觀表。

使用 IAM 政策管理資料共用 API 操作的存取

若要控制資料共用 API 操作的存取權，請使用 IAM 動作型政策。如需有關管理 IAM 政策的詳細資訊，請參閱《IAM 使用者指南》中的理 IAM 政策。

如需使用資料共用 API 操作所需許可的相關資訊，請參閱《Amazon Redshift 管理指南》中的使用資料共用 API 操作所需的權限。

若要讓跨帳戶資料共用更安全，您可以使用條件式金鑰 `ConsumerIdentifier` 進行 `AuthorizeDataShare` 和 `DeauthorizeDataShare` API 操作。這樣，您可以明確控制哪些 AWS 帳戶 可以對兩個 API 操作進行調用。

對於不是您自己帳戶的取用者，您可以拒絕授權或取消授權資料共用。若要這麼做，請在 IAM 政策中指定 AWS 帳戶 數字。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "VisualEditor0",
            "Effect": "Deny",
            "Action": [
                "redshift:AuthorizeDataShare",
                "redshift:DeauthorizeDataShare"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "redshift:ConsumerIdentifier": "555555555555"
                }
            }
        }
    ]
}
```

您可以允許具有 a 的生產者明確地與 IAM 政策中具有 111122223333 AWS 帳戶 的消費者共享。DataShareArn **`testshare2`**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "redshift:AuthorizeDataShare",
                "redshift:DeauthorizeDataShare"
            ],
            "Resource": "arn:aws:redshift:us-
east-1:666666666666:datashare:af06285e-8a45-4ee9-b598-648c218c8ff1/testshare2",
            "Condition": {
                "StringEquals": {
                    "redshift:ConsumerIdentifier": "111122223333"
                }
            }
        }
    ]
}
```

查詢資料庫

在 Amazon Redshift 中存取共用資料

您可以使用標準 SQL 介面、JDBC 或 ODBC 驅動程式及資料 API 來探索共用資料。您也可以透過熟悉的商業智慧 (BI) 和分析工具，以高效能方式查詢資料。您可以參考其他 Amazon Redshift 資料庫中的物件來執行查詢，只要這些資料庫是您有權存取的叢集的本機和遠端資料庫即可。

只要保持連線至叢集中的本機資料庫，就可以這麼做。然後，您可以從資料庫建立取用者資料庫以使用共用資料。

完成此操作之後，您就可以執行聯結資料集的跨資料庫查詢。您可以使用 3 部分表示法 (*consumer_database_name*.*schema_name*.*table_name*) 查詢取用者資料庫中的物件。您也可以使用取用者資料庫中結構描述的外部結構描述連結進行查詢。您可以同時查詢本機資料和相同查詢內其他叢集共用的資料。這類查詢可以參考來自目前連線資料庫和其他未連線資料庫的物件，包括從資料共用建立的取用者資料庫。

在 Amazon Redshift 中存取資料共用的中繼資料

為了協助叢集管理員探索資料共用，Amazon Redshift 提供了一組中繼資料檢視來列出資料共用。這些檢視會列出叢集中建立的資料共用，以及從相同帳戶內的其他叢集、其他帳戶或其他 AWS 區域接收的資料共用。這些檢視會顯示下列資訊：

- 叢集共用和接收的資料共用
- 資料共用中資料庫物件的內容，包括基本共用中繼資料、物件和取用者

使用 SVV_DATASHARES 可檢視叢集中建立的所有資料共用清單 (輸出)，以及其他人共用的所有資料共用 (輸入)。如需詳細資訊，請參閱 SVV_DATASHARES。

使用 SVV_DATASHARE_CONSUMERS 來檢視資料取用者清單。如需詳細資訊，請參閱 SVV_DATASHARE_CONSUMERS。

使用 SVV_DATASHARE_OBJECTS 可檢視叢集中建立的所有資料共用的物件清單 (輸出)，以及其他人共用的所有資料共用的物件清單 (輸入)。如需詳細資訊，請參閱 SVV_DATASHARE_OBJECTS。

將 Amazon Redshift 資料共用與商業智慧工具整合

若要整合資料共用與商業智慧 (BI) 工具，我們建議您使用 Amazon Redshift JDBC 或 ODBC 驅動程式。

Amazon Redshift JDBC 和 ODBC 驅動程序支援驅動程式中的 GetCatalogs API 操作，該操作會傳回所有資料庫的清單，包括從資料共用建立的資料庫。這些驅動程式也支援下游作業，例如 GetSchemas、GetTables 等，這些作業會從 GetCatalogs 傳回的所有資料庫傳回資料。即使呼叫中未明確指定目錄，驅動程式也會提供此支援。如需 JDBC 或 ODBC 驅動程式的相關資訊，請參閱《Amazon Redshift 管理指南》中的在 Amazon Redshift 中設定連線。

您不能直接連線到從資料庫中建立的取用者資料庫。連線到叢集上的本機資料庫。如果您的工具中有連線切換使用者介面，資料庫清單應該只會包含本機叢集資料庫。清單應該排除從資料共用建立的取用者資料庫，以提供最佳體驗。您可以使用 SVV_REDSHIFT_DATABASE 檢視中的選項來篩選資料庫。

在 Amazon Redshift 中監視和稽核資料共用

透過稽核資料共用，生產者可以追蹤資料共用的演變。例如，稽核有助於追蹤何時建立資料庫、新增或移除物件，以及授與或撤銷 Amazon Redshift 叢集、 AWS 帳戶或區域的許可。 AWS

除了稽核之外，生產者和取用者還可以追蹤各種細微性 (例如帳戶、叢集和物件層級) 的資料共用使用情況。如需追蹤使用情形和稽核檢視的相關資訊，請參閱 SVL_DATASHARE_CHANGE_LOG 和 SVL_DATASHARE_USAGE_PRODUCER。

您可以透過查詢系統檢視來監視資料共用。

1. 想要共用資料的生產者叢集管理員會建立 Amazon Redshift 資料共用。然後，生產者叢集管理員會新增所需的資料庫物件。這些可能是資料共用的結構描述、資料表和檢視，並且會指定要與之共用物件的取用者清單。

   使用下列系統檢視來查看合併的檢視，以追蹤生產者和/或用戶叢集上資料共用的變更和使用情況：

   - SYS_DATASHARE_CHANGE_LOG
   - SYS_DATASHARE_USAGE_CONSUMER
   - SYS_DATASHARE_USAGE_PRODUCER

   使用下列系統檢視來查看輸出資料共用的資料共用物件和資料取用者資訊：

   - SVV_DATASHARES
   - SVV_DATASHARE_CONSUMERS
   - SVV_DATASHARE_OBJECTS
2. 取用者叢集管理員會查看他們被授權使用的資料庫，並使用 SVV_DATASHARES 檢視輸入資料共用來檢閱每個資料共用的內容。

若要使用共用的資料，每個取用者叢集管理員會從資料共用中建立 Amazon Redshift 資料庫。然後，管理員會將許可指派給取用者叢集中的適當使用者和角色。使用者和角色可以透過檢視下列中繼資料系統檢視，將共用物件列為標準中繼資料查詢的一部分，並且可以立即開始查詢資料。

- SVV_REDSHIFT_COLUMNS
- SVV_REDSHIFT_DATABASES
- SVV_REDSHIFT_FUNCTIONS
- SVV_REDSHIFT_SCHEMAS
- SVV_REDSHIFT_TABLES

若要檢視 Amazon Redshift 本機和共用結構描述及外部結構描述的物件，請使用下列中繼資料系統檢視進行查詢。

- SVV_ALL_COLUMNS
- SVV_ALL_SCHEMAS
- SVV_ALL_TABLES

整合 Amazon Redshift 資料共用 AWS CloudTrail

數據共享與集成 AWS CloudTrail。 CloudTrail 這是一項服務，可提供使用者、角色或 Amazon Redshift 中的 AWS 服務所採取的動作記錄。 CloudTrail 擷取所有 API 呼叫，以便將資料共用為事件。擷取的呼叫包括來自 AWS CloudTrail 主控台的呼叫，以及對資料共用作業的程式碼呼叫。如需與 Amazon Redshift 整合的詳細資訊 AWS CloudTrail，請參閱使用 CloudTrail.

如需相關資訊 CloudTrail，請參閱如何 CloudTrail 運作。

管理資料共用工作

您可以透過使用 SQL 介面或 Amazon Redshift 主控台開始使用資料共用功能。

主題

- 使用 SQL 介面管理資料共用
- 使用主控台管理資料共用
- 使用 AWS CloudFormation管理資料共用
- 使用主控台透過寫入功能管理資料共用 (預覽)

使用 SQL 介面管理資料共用

您可以在 AWS 帳戶內部或 AWS 區域之間的不同 Amazon Redshift 叢集之間共用資料以供讀取。

主題

共用資料的讀取存取權限 AWS 帳戶

您可以在 AWS 帳戶內部的不同 Amazon Redshift 叢集之間共用資料以供讀取。

以生產者叢集管理員或資料庫擁有者的身分共用資料以供讀取

1. 在叢集中建立資料共用。如需詳細資訊，請參閱 CREATE DATASHARE。

   ```
   CREATE DATASHARE salesshare;
   ```

   叢集超級使用者和資料庫擁有者可以建立資料共用。每個資料共用都會在建立期間與資料庫相關聯。只有來自該資料庫的物件才能在該資料共用中共用。可以在具有相同或不同物件細微程度的相同資料庫上建立多個資料共用。叢集可以建立的資料共用數量不限。

   您也可以使用 Amazon Redshift 主控台建立資料共用。如需詳細資訊，請參閱 建立資料共用。
2. 委派在資料共用上操作的許可。如需詳細資訊，請參閱 GRANT 或 REVOKE 。

   下列範例政策會授予許給 `salesshare` 上的 `dbuser`。

   ```
   GRANT ALTER, SHARE ON DATASHARE salesshare TO dbuser;
   ```

   叢集超級使用者和資料共用的擁有者可以對其他使用者授與或撤銷資料共用的修改許可。
3. 在資料庫共用中新增物件或移除物件。若要將物件新增至資料共用，請在新增物件之前先新增結構描述。當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。務必明確地新增這些物件。如需詳細資訊，請參閱 ALTER DATASHARE。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

您也可將檢視新增到資料共用。

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM
 public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;
```

使用 ALTER DATASHARE 來共用結構描述及指定結構描述中的資料表、檢視和函數。超級使用者、資料共用擁有者或在資料共用上擁有 ALTER 或 ALL 許可的使用者可以修改資料共用，以在其中新增或移除物件。使用者應具有在資料共用中新增或移除物件的許可。使用者也必須是物件的擁有者，或具有物件的 SELECT、USAGE 或 ALL 許可。

您也可以使用 GRANT 將物件加入至資料清單。此範例顯示如何：

```
GRANT SELECT ON TABLE public.tickit_sales_redshift TO DATASHARE salesshare;
```

此語法在功能上等同於`ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;`。

使用 INCLUDENEW 子句將指定結構描述中建立的任何新資料表、檢視或 SQL 使用者定義函數 (UDF) 新增至資料共用。只有超級使用者可以為每個「資料共用-結構描述」配對修改此屬性。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除物件。如需詳細資訊，請參閱將資料共用物件新增到資料共用、從資料共用中移除資料共用物件及編輯在您帳戶中建立的資料共用。

4. 在資料共用中新增或移除取用者。下列範例會將取用者叢集命名空間新增至 `salesshare`。命名空間是帳戶中取用者叢集的命名空間全域唯一識別碼 (GUID)。如需詳細資訊，請參閱 GRANT 或 REVOKE 。

```
GRANT USAGE ON DATASHARE salesshare TO NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

在 GRANT 陳述式中，您只能將許可授予一個資料共用取用者。

叢集超級使用者和資料共用物件的擁有者或在資料共用上擁有 SHARE 許可的使用者，都可以在資料共用中新增或移除將取用者。為了這麼做，他們會使用 GRANT USAGE 或 REVOKE USAGE。

若要尋找目前所見之叢集的命名空間，您可以使用 SELECT CURRENT_NAMESPACE 命令。若要尋找同一叢集中不同叢集的命名空間 AWS 帳戶，請前往 Amazon Redshift 主控台叢集詳細資訊頁面。在該頁面上，找到新增的命名空間欄位。

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除資料取用者。如需詳細資訊，請參閱 將資料取用者新增至資料共用 及 從資料共用中移除資料取用者。

5. (選擇性) 將安全性限制新增至資料共用。下列範例顯示具有公用 IP 存取權的取用者叢集可以讀取資料共用。如需詳細資訊，請參閱 ALTER DATASHARE。

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE = TRUE;
```

您可以在建立資料共用之後修改有關取用者類型的屬性。例如，您可以定義要使用指定資料共用中資料的叢集無法公開存取。若查詢來自不符合資料共用中所指定安全性限制的取用者叢集，則會在查詢執行期遭到拒絕。

您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱 編輯在您帳戶中建立的資料共用。

6. 列出在叢集中建立的資料共用，並查看資料共用的內容。

下列範例會顯示名為 salesshare 之資料共用的資訊。如需詳細資訊，請參閱 DESC DATASHARE 及 SHOW DATASHARES。

```
DESC DATASHARE salesshare;


 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name          |   include_new
-------------------+--------------------------------------+------------
+------------+-------------+---------------------------------+-------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_users_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_venue_redshift   |
```

```
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_category_redshift|
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_date_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_event_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_listing_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_sales_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | schema      | public                         |  t
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | view        | public.sales_data_summary_view |
```

下列範例顯示生產者叢集上的輸出資料共用。

```
SHOW DATASHARES LIKE 'sales%';
```

輸出結果類似如下。

```
share_name | share_owner  | source_database | consumer_database | share_type |
 createdate      | is_publicaccessible  | share_acl | producer_account |
 producer_namespace
-----------+-------------+-----------------+-------------------+------------
+---------------------+----------------------+-----------+------------------
+--------------------------------------
salesshare |    100       | dev             |                   |  OUTBOUND
 | 2020-12-09 02:27:08 |          True        |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

如需詳細資訊，請參閱 DESC DATASHARE 及 SHOW DATASHARES。

您也可以使用 SVV_DATASHARES、SVV_DATASHARE_CONSUMERS 和 SVV_DATASHARE_OBJECTS 來檢視資料共用、資料共用中的物件，以及資料共用取用者。

7. 捨棄資料共用。如需詳細資訊，請參閱 DROP DATASHARE。

   您可以在任何時間點使用 DROP DATASHARE 刪除資料共用物件。叢集超級使用者和資料共用擁有者可以捨棄資料共用。

   以下範例會捨棄名為 salesshare 的資料共用。

```
DROP DATASHARE salesshare;
```

您也可以使用 Amazon Redshift 主控台刪除資料共用。如需詳細資訊，請參閱 刪除在您帳戶中建立的 資料共用。

8. 使用 ALTER DATASHARE 隨時從資料共用中點移除物件。使用 REVOKE USAGE ON 來撤銷特定取用者在資料共用上的許可。它會撤銷資料共用中物件的 USAGE 許可，並立即停止對所有取用者叢集的存取。存取權被撤銷後，列出資料共用和中繼資料查詢 (例如列出資料庫和資料表) 不會傳回共用物件。

```
ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
```

您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱 編輯在您帳戶中建立的資料共用。

9. 如果您不想再與取用者共用資料，請撤銷命名空間中的資料共用存取權。

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱 編輯在您帳戶中建立的資料共用。

以取用者叢集管理員的身分共用資料以供讀取

1. 列出可供您使用的資料共用，並檢視資料共用的內容。如需詳細資訊，請參閱 DESC DATASHARE 及 SHOW DATASHARES。

   下列範例會顯示指定生產者命名空間的輸入資料共用資訊。當您以取用者叢集管理員身分執行 DESC DATASHARE 時，您必須指定 NAMESPACE 選項以檢視輸入資料共用。

```
DESC DATASHARE salesshare OF NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';


 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name           |   include_new
-------------------+--------------------------------------+------------
+------------+-------------+----------------------------------+------------------
```

```
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_users_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_venue_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_category_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_date_redshift     |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_event_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_listing_redshift  |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_sales_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | schema      | public                          |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | view        | public.sales_data_summary_view  |
```

只有叢集超級使用者才能執行此動作。您也可以使用 SVV_DATASHARES 來檢視資料共用，以及使用 SVV_DATASHARE_OBJECTS 來檢視資料共用內的物件。

下列範例顯示取用者叢集中的輸入資料共用。

```
SHOW DATASHARES LIKE 'sales%';


 share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
------------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+------------------
+--------------------------------------
 salesshare |             |                 |                   | INBOUND
    |            |         t           |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

2. 身為資料庫超級使用者，您可以建立參照資料共用的本機資料庫。如需詳細資訊，請參閱 CREATE DATABASE。

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

如果您想要對本機資料庫中物件的存取進行更精細的控制，請在建立資料庫時使用 WITH PERSOMMENTS 子句。這可讓您在步驟 4 中為資料庫中的物件授予物件層級權限。

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

您可以透過查詢 SVV_REDSHIFT_DATABASES 檢視來查看您從資料共用中建立的資料庫。您無法連線到從資料共用中建立的這些資料庫，這些資料庫僅限讀取。不過，您可以連線到取用者叢集上的本機資料庫，並執行跨資料庫查詢，從資料共用建立的資料庫中查詢資料。您不能在現有資料共用中建立的資料庫物件頂端建立資料共用。不過，您可以將資料複製到取用者叢集上的個別資料表中，執行所需的任何處理，然後共用已建立的新物件。

您也可以使用 Amazon Redshift 主控台從資料共用中建立資料共用。如需詳細資訊，請參閱 從資料共用中建立資料庫。

3. (選擇性) 建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
 'public';
```

4. 視需要，將從資料共用建立的資料庫和結構描述參照的許可授予使用者叢集中的使用者和角色。如需詳細資訊，請參閱 GRANT 或 REVOKE 。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

如果您在沒有 WITH PERSOMMENCE 的情況下建立資料庫，則只能將從資料共用建立的整個資料庫的權限指派給您的使用者和角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。如果是這樣，您可以建立指向資料共用中特定結構描述的外部結構描述參考 (如上一個步驟所述)，並在結構描述層級上提供精細的許可。

您也可以在共用物件之上建立近期繫結檢視，並使用這些檢視來指派精細的許可。也可以考慮讓生產者叢集以所需的精細程度為您建立其他資料共用。

如果您在步驟 2 中使用 WITH PERMISSIONS 建立資料庫，則必須為共用資料庫中的物件指派物件層級權限。只有 USAGE 權限的使用者在獲得額外的物件層級權限之前，不能存取使用 WITH PERMISSIONS 建立的資料庫中的任何物件。

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

5. 查詢資料共用中共用物件的資料。

   在取用者資料庫和取用者叢集結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集中的本機物件。若要做到這一點，他們會使用 JDBC 或 ODBC 驅動程式或 SVV_ALL 和 SVV_REDSHIFT 檢視。

   生產者叢集在每個結構描述中可能會有多個資料庫、資料表和檢視的結構描述。取用者端的使用者只能看到透過資料共用提供的物件子集。這些使用者無法從生產者叢集看到整個中繼資料。此方法有助於透過資料共用提供精細的中繼資料安全控制

   您可以繼續連線到本機叢集資料庫。但是現在，您也可以使用三部分 database.schema.table 表示法，從資料共用中建立的資料庫和結構描述中讀取。您可以執行跨越任何和所有可見資料庫的查詢。這些資料庫可以是叢集上的本機資料庫，也可以是從資料共用建立的資料庫。取用者叢集無法連線至從資料共用建立的資料庫。

   您可以使用完整資格存取資料。如需詳細資訊，請參閱 使用跨資料庫查詢的範例。

```
SELECT * FROM sales_db.public.tickit_sales_redshift ORDER BY 1,2 LIMIT 5;

 salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime
---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------
       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |    728.00 |
   109.20 | 2008-02-18 02:36:48
       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |     76.00 |
    11.40 | 2008-06-06 05:00:16
       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |    350.00 |
    52.50 | 2008-06-06 08:26:17
       4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |    175.00 |
    26.25 | 2008-06-09 08:38:52
       5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |    154.00 |
    23.10 | 2008-08-31 09:17:02
```

您只能在共用物件上使用 SELECT 陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

除了查詢之外，取用者還可以在共用物件上建立檢視。僅支援近期繫結檢視或具體化視觀表。Amazon Redshift 不支援共用資料的一般檢視。取用者建立的檢視可跨越多個本機資料庫或從資料共用建立的資料庫。如需詳細資訊，請參閱 CREATE VIEW。

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

共用資料的寫入權限 (預覽)

您可以跨不同 Amazon Redshift 叢集或 Amazon Redshift 無伺服器工作群組共用讀取和寫入的資料庫物件 AWS 帳戶，也可以跨帳戶和跨區域共用相同的 Amazon Redshift 無伺服器工作群組。本主題中的程序說明如何設定包含寫入權限的資料共用。您可以針對不同的資料表授與權限，例如 SELECT、INSERT 和 UPDATE，以及架構的「使用」和「建立」。一旦確認寫入交易，資料即可供所有倉儲使用。生產者帳戶管理員可以決定特定的命名空間或區域是否獲得唯讀 read-and-write，還是對資料的任何存取權。

以下各節說明如何設定資料共用。這些過程假設您正在佈建的叢集或 Amazon Redshift Serverless 工作群組中的資料庫中工作。

唯讀資料共用與讀寫資料共用

之前資料共用中的物件在所有情況下都是唯讀的。寫入資料共用中的物件是一項新功能。只有當生產者特別授予寫入權限 (如 INSERT 或 CREATE) 給資料共用的物件時，資料共用中的物件才能啟用寫入功能。此外，對於跨帳戶共享，生產者必須授權寫入的數據存儲，並且消費者必須關聯特定的叢集和工作群組以進行寫入。本主題後續章節中的詳細資訊如下。

您可以授予資料庫的權限 (預覽)

您可以在資料共用內容中授予不同的物件類型和各種權限。

結構描述：

- USAGE
- CREATE

資料表：

- SELECT
- INSERT
- UPDATE
- DELETE
- TRUNCATE
- DROP
- REFERENCES

函數：

- EXECUTE

資料庫：

- CREATE

預覽版資料共用的要求和限制

- 連線 — 您必須直接連線至資料清理資料庫，或執行 USE 指令來寫入資料存放區。不過我們很快將能夠透過三部分記號來實現這項操作。
- 可用性 — 您必須使用無伺服器工作群組、ra3.4xl 叢集或 ra3.16xl 叢集才能使用此功能。已規劃為 ra3.xlplus 叢集提供支援。
- 中繼資料探索 — 當您是透過 Redshift JDBC、ODBC 或 Python 驅動程式直接連線至資料存取資料庫的取用者時，您可以透過下列方式檢視目錄資料：
  - SQL SHOW 命令。
  - 查詢 information_schema 資料表和視觀表。
  - 查詢 SVV 中繼資料檢視。

- 資料 API — 您無法透過資料 API 連線至資料清單資料庫。即將提供相關支援。
- 權限可見性 — 取用者看不到授與資料庫的權限。我們很快就會新增這項功能。
- 加密 — 對於跨帳戶資料共用，生產者和消費者叢集都必須加密。
- 隔離層級 — 資料庫的隔離層級必須是快照隔離，才能允許其他無伺服器工作群組和叢集寫入資料庫。
- auto 操作 — 寫入資料清單物件的消費者不會觸發自動分析操作。因此，生產者必須在將資料插入資料表後手動執行分析，才能更新資料表統計資料。如果沒這麼做，查詢計劃可能不是最好的。
- 多重陳述式查詢和交易 — 目前不支援交易區塊以外的多重陳述式查詢。因此，如果您使用像 dbeaver 這樣的查詢編輯器，並且有多個寫入查詢，則需要將查詢包裝在明確的 BEGIN... END 事務語句中。

支援的 SQL 陳述式

以下陳述式適用於透過寫入進行資料共用的公開預覽版本：

- BEGIN | START TRANSACTION
- END | COMMIT | ROLLBACK
- COPY 不含 COMPUPDATE
- { CREATE | DROP } SCHEMA
- { CREATE | DROP | SHOW } TABLE
- CREATE TABLE table_name AS
- DELETE
- { GRANT | REVOKE } privilege_name ON OBJECT_TYPE object_name TO consumer_user
- INSERT
- SELECT
- INSERT INTO SELECT
- TRUNCATE
- UPDATE
- 超級資料類型列欄

不支援的陳述式類型 — 不支援下列項目：

- 寫入生產者時，對取用者倉庫進行多陳述式查詢。
- 並行擴縮查詢從取用者寫入生產者。
- 從取用者寫入到生產者的自動複製作業。
- 從取用者到生產者的串流工作寫入。
- 在生產者叢集上建立零 ETL 整合表的取用者。如需零 ETL 整合的相關資訊，請參閱使用零 ETL 整合。
- 使用交錯排序索引鍵寫入資料表。

以製作者帳號管理員身分擁有寫入權限的帳戶內共用資料 (預覽)

之前資料共用中的物件在所有情況下都是唯讀的。寫入資料共用中的物件是一項新功能。只有當生產者特別授予寫入權限 (如 INSERT 或 CREATE) 給資料共用的物件時，資料共用中的物件才能啟用寫入功能。本主題後續章節中的詳細資訊如下。

如果您正在尋找唯讀資料共用的現有文件，請參閱 Amazon Redshift 中的跨叢集共用資料。

若要開始資料共用，生產者上的管理員會建立資料共用，並將物件加入至其中：

1. 生產者資料庫擁有者或超級使用者會建立資料共用。資料共用是資料庫物件、權限和取用者的邏輯容器。(取用者是您帳戶和其他帳戶中的叢集或 Amazon Redshift Serverless 命名空間。) 每個資料共用都與其建立的資料庫相關聯，只能新增來自該資料庫的物件。以下命令會建立資料共用。

   ```
   CREATE DATASHARE my_datashare [PUBLICACCESSIBLE = TRUE];
   ```

   設定 PUBLICACCESSIBLE = TRUE 可讓取用者從可公開存取的叢集和已佈建的工作群組查詢您的資料共用。如果您不允許，請將其忽略，或明確地將其設定為 false。

   資料共用所有者必須授予他們想要新增到資料共用的結構描述的 USAGE。GRANT 命令是新的。它用於授予結構描述上的各種操作，包括 CREATE 和 USAGE.用。結構描述包含共用物件：

   ```
   CREATE SCHEMA myshared_schema1;
   CREATE SCHEMA myshared_schema2;

   GRANT USAGE ON SCHEMA myshared_schema1 TO DATASHARE my_datashare;
   GRANT CREATE, USAGE ON SCHEMA myshared_schema2 TO DATASHARE my_datashare;
   ```

   或者，系統管理員也可以繼續執行 ALTER 命令，將結構描述新增至資料共用。以這種方式新增結構描述時，只會授予 USAGE 權限。

```
ALTER DATASHARE my_datashare ADD SCHEMA myshared_schema1;
```

2. 管理員新增結構描述之後，就可以授予結構描述中物件的資料共用權限。這些是讀取和寫入權限 GRANT ALL 範例顯示如何授予所有權限。

```
GRANT SELECT, INSERT ON TABLE myshared_schema1.table1, myshared_schema1.table2,
 myshared_schema2.table1
TO DATASHARE my_datashare;

GRANT ALL ON TABLE myshared_schema1.table4 TO DATASHARE my_datashare;
```

   您可以繼續執行像 ALTER DATASHARE 這樣的命令來新增資料表。當您這麼做時，只會授予新增物件的 SELECT 權限。

```
ALTER DATASHARE my_datashare ADD TABLE myshared_schema1.table1,
 myshared_schema1.table2, myshared_schema2.table1;
```

3. 系統管理員會將資料共用的使用權授予帳戶中的特定命名空間。您可以在叢集詳細資料頁面、Amazon Redshift Serverless 命名空間詳細資料頁面，或藉由執行命令 SELECT current_namespace;，來尋找命名空間 ID 以做為 ARN 的一部分。如需詳細資訊，請參閱 CURRENT_NAMESPACE前。

```
GRANT USAGE ON DATASHARE my_datashare TO NAMESPACE '86b5169f-012a-234b-9fbb-
e2e24359e9a8';
```

跨帳戶共用資料的寫入權限 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如果您尚未在 PREVIEW_2023 軌道上創建數據保護，請轉到共享對數據的寫入訪問權限（預覽）開始使用。

將共用資料關聯為取用者資料安全管理員 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如果您尚未在 PREVIEW_2023 軌道上創建數據保護，請轉到共享對數據的寫入訪問權限（預覽）開始使用。

先決條件：在生產者管理員授予共用資料庫物件的特定動作之後執行本節所述步驟，如果資料共用正在與其他帳戶共用，則生產者安全管理員會授予存取權。

取用者安全管理員確定以下內容：

- 帳戶中的所有命名空間、帳戶中特定區域的命名空間或特定命名空間，是否都可以存取資料共用。
- 如果命名空間可以存取資料共用，無論這些命名空間是否具有寫入權限。

取用者安全管理員可以透過主控台、CLI 或透過 API 建立資料共用關聯。如果是 CLI，系統管理員會使用下列命令：

```
associate-data-share-consumer
--data-share-arn <value>
--consumer-identifier <value>
[--allow-writes | --no-allow-writes]
```

如需有關命令的詳細資訊，請參閱 associate-data-share-consumer。

當資料共用與命名空間相關聯時，取用者安全管理員必須明確將 `allow-writes` 設定為 true，以允許使用 INSERT 和 UPDATE 命令。如果不這樣做，使用者只能執行讀取作業，例如 SELECT、USAGE 或 EXECUTE 權限。

您可以透過使用不同的值再次呼叫 `associate-data-share-consumer`，以變更資料共用命名空間的關聯。舊關聯會被新關聯覆寫，因此，如果您最初有建立關聯並設定 `allow-writes`，但建立關聯並指定 `no-allow-writes`，或者只是沒有指定值，則會撤銷取用者的寫入權限。

以生產者安全管理員身分授權資料共用以進行寫入 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如果您尚未在 PREVIEW_2023 軌道上創建數據保護，請轉到共享對數據的寫入訪問權限（預覽）開始使用。

> **Note**
>
> 這僅適用於在帳戶之間共用資料共用時。

生產者安全管理員確定以下內容：

- 是否其他帳戶可以存取資料共用功能。
- 如果某帳戶可以存取資料共用，無論該帳戶是否具有寫入權限。

授權資料共用所需的下列 IAM 權限：

紅移：AuthorizeData分享

您可以使用 CLI 呼叫或 API 授權使用和寫入：

```
authorize-data-share
--data-share-arn <value>
--consumer-identifier <value>
[--allow-writes | --no-allow-writes]
```

如需有關命令的詳細資訊，請參閱 authorize-data-share。

取用者識別碼可以是：

- 一個十二位數的 AWS 帳戶 ID。
- 命名空間標識符 ARN。

請注意，在授權步驟不會授予寫入權限。授權資料共用以進行寫入，這只允許帳戶擁有資料共用管理員授予的寫入權限。如果管理員不允許寫入，則特定取用者可用的唯一權限是 SELECT、USAGE 和 EXECUTE。

您可以透過再次呼叫 `authorize-data-share` 來變更資料共用取用者的授權，但使用的是不同的值。舊授權會被新授權覆寫。因此，如果您最初授權並允許寫入，但重新授權並指定 `no-allow-writes` 或根本不指定值，則取用者將會撤銷其寫入權限。

提供資料共用的區域 (預覽)

> 這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如果您尚未在 PREVIEW_2023 軌道上創建數據保護，請轉到共享對數據的寫入訪問權限（預覽）開始使用。

下列區域提供預覽版的資料共用：

- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國東部 (俄亥俄) (us-east-2)
- 美國西部 (奧勒岡) (us-west-2)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

共用資料 AWS 帳戶

您可以在 AWS 帳戶之間共用資料以供讀取。跨 AWS 帳戶 工作共用資料類似於在帳戶內共用資料。不同之處在於，在 AWS 帳戶之間共用資料時需要雙向交握。生產者帳戶管理員可以授權取用者帳戶存取資料共用，也可以選擇不授權任何存取權。若要使用授權的資料共用，取用者帳戶管理員可以與資料共用建立關聯。系統管理員可以將資料保護與整個 AWS 帳戶 或與消費者帳戶中的特定叢集建立關聯，或拒絕資料保留。如需在帳戶內共用資料的相關資訊，請參閱 共用資料的讀取存取權限 AWS 帳戶。

資料共用的資料取用者可以是相同帳戶中或不同 AWS 帳戶中的叢集命名空間。您不需要建立單獨的資料共用，即可在帳戶中和帳戶之間共用。

對於跨帳戶資料共用，生產者和取用者叢集都必須加密。

與共用資料時 AWS 帳戶，生產者叢集管理員會以實體的 AWS 帳戶 身分與共用。取用者叢集管理員可以決定取用者帳戶中的哪些叢集命名空間可以存取資料共用。

主題

- 生產者叢集管理員動作
- 取用者帳戶管理員動作
- 取用者叢集管理員動作

生產者叢集管理員動作

如果您是生產者叢集管理員或資料庫擁有者，請依照下列步驟執行：

1. 在叢集中建立資料共用，並將資料共用物件新增至資料共用。如需如何建立資料共用和將資料共用物件新增至資料共用的詳細步驟，請參閱 共用資料的讀取存取權限 AWS 帳戶。如需有關 CREATE DATASHARE 和 ALTER DATASHARE 的資訊，請參閱 CREATE DATASHARE 和 ALTER DATASHARE。

   下列範例會將不同的資料共用物件加入至資料共用 salesshare。

   ```
   -- Add schema to datashare
   ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;

   -- Add table under schema to datashare
   ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;

   -- Add view to datashare
   ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;

   -- Add all existing tables and views under schema to datashare (does not include
    future table)
   ALTER DATASHARE salesshare ADD ALL TABLES in schema public;
   ```

   您也可以使用 Amazon Redshift 主控台建立或編輯資料共用。如需詳細資訊，請參閱 建立資料共用及 編輯在您帳戶中建立的資料共用。
2. 委派在資料共用上操作的許可。如需詳細資訊，請參閱 GRANT 或 REVOKE 。

   下列範例政策會授予許給 salesshare 上的 dbuser。

```
GRANT ALTER, SHARE ON DATASHARE salesshare TO dbuser;
```

叢集超級使用者和資料共用的擁有者可以對其他使用者授與或撤銷資料共用的修改許可。

3. 在資料共用中新增或移除取用者。下列範例會將 AWS 帳戶 ID 新增至 salesshare。如需詳細資訊，請參閱 GRANT 或 REVOKE 。

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012';
```

在 GRANT 陳述式中，您只能將許可授予一個資料取用者。

叢集超級使用者和資料共用物件的擁有者或在資料共用上擁有 SHARE 許可的使用者，都可以在資料共用中新增或移除將取用者。為了這麼做，他們會使用 GRANT USAGE 或 REVOKE USAGE。

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除資料取用者。如需詳細資訊，請參閱 將資料取用者新增至資料共用 及 從資料共用中移除資料取用者。

4. （可選）AWS 帳戶 如果您不想再與消費者共享數據，請撤消對數據保護的訪問權限。

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '123456789012';
```

如果您是生產者帳戶管理員，請依照下列步驟執行：

將使用授與之後 AWS 帳戶，資料清單狀態為。pending_authorization生產者帳戶管理員應該使用 Amazon Redshift 主控台授權資料共用，並選擇資料取用者。

請登入以下網址：https://console.aws.amazon.com/redshiftv2/ 然後選擇要授權存取資料共用或從中移除授權的資料取用者。獲授權的資料取用者會收到對資料共用採取動作的通知。如果您要將叢集命名空間新增為資料取用者，則不必執行授權。授權資料取用者之後，他們就可以存取資料共用物件，並建立取用者資料庫來查詢資料。如需詳細資訊，請參閱 授予或移除資料共用的授權。

取用者帳戶管理員動作

如果您是取用者帳戶管理員，請依照下列步驟執行：

若要將從其他帳戶共用的一或多個資料庫與帳戶中的整個 AWS 帳戶 或特定叢集命名空間建立關聯，請使用 Amazon Redshift 主控台。

請登入以下網址：https://console.aws.amazon.com/redshiftv2/ 然後，將從其他帳戶共用的一或多個資料庫與帳戶中的整個 AWS 帳戶 或特定叢集命名空間建立關聯。如需詳細資訊，請參閱 建立資料共用的關聯。

關聯 AWS 帳戶 或特定叢集命名空間之後，資料庫就可供使用。您也可以隨時變更資料共用關聯。將個別叢集命名空間的關聯變更為時 AWS 帳戶，Amazon Redshift 會以資訊覆寫叢集命名空間。 AWS 帳戶 將關聯從某個叢集命名空間變更 AWS 帳戶 為特定叢集命名空間時，Amazon Redshift 會以叢集命名空間 AWS 帳戶 資訊覆寫資訊。帳戶中的所有叢集命名空間都可以存取資料。

取用者叢集管理員動作

如果您是取用者叢集管理員，請依照下列步驟執行：

1. 列出可供您使用的資料共用，並檢視資料共用的內容。只有當生產者叢集管理員已授權資料共用，且取用者叢集管理員已接受資料共用並與之建立關聯時，才能使用資料共用的內容。如需詳細資訊，請參閱 DESC DATASHARE 及 SHOW DATASHARES。

   下列範例會顯示指定生產者命名空間的輸入資料共用資訊。當您以取用者叢集管理員身分執行 DESC DATAHSARE 時，您必須指定 NAMESPACE 和帳戶 ID 以檢視輸入資料共用。對於輸出資料共用，請指定資料共用名稱。

```
SHOW DATASHARES LIKE 'sales%';

share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
-----------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+------------------
+--------------------------------------
salesshare |             |                 |                   | INBOUND    |
            |        t            |           | 123456789012    | 'dd8772e1-
d792-4fa4-996b-1870577efc0d'
```

```
DESC DATASHARE salesshare OF ACCOUNT '123456789012' NAMESPACE 'dd8772e1-
d792-4fa4-996b-1870577efc0d';


 producer_account |          producer_namespace          | share_type | share_name |
 object_type |           object_name
------------------+--------------------------------------+------------+------------
+-------------+---------------------------------
```

```
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_users_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_venue_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_category_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_date_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_event_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_listing_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 table       | public.tickit_sales_redshift
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND    | salesshare |
 schema      | public
(8 rows)
```

只有叢集超級使用者才能執行此動作。您也可以使用 SVV_DATASHARES 來檢視資料共用，以及使用 SVV_DATASHARE_OBJECTS 來檢視資料共用內的物件。

下列範例顯示取用者叢集中的輸入資料共用。

```
SELECT * FROM SVV_DATASHARES WHERE share_name LIKE 'sales%';

share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
-----------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+------------------
+--------------------------------------
salesshare |             |                 |                   | INBOUND    |
            |        t            |           | 123456789012      | 'dd8772e1-
d792-4fa4-996b-1870577efc0d'
```

```
SELECT * FROM SVV_DATASHARE_OBJECTS WHERE share_name LIKE 'sales%';
 share_type | share_name | object_type |           object_name           |
 producer_account |          producer_namespace
------------+------------+-------------+---------------------------------
+------------------+--------------------------------------
 INBOUND    | salesshare | table       | public.tickit_users_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
```

```
 INBOUND    | salesshare | table       | public.tickit_venue_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_category_redshift |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_date_redshift     |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_event_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_listing_redshift  |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | table       | public.tickit_sales_redshift    |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
 INBOUND    | salesshare | schema      | public                          |
 123456789012     | dd8772e1-d792-4fa4-996b-1870577efc0d
(8 rows)
```

2. 建立參照資料共用的本機資料庫。從資料共用建立資料庫時，請指定 NAMESPACE 和帳戶 ID。如需詳細資訊，請參閱 CREATE DATABASE。

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF ACCOUNT '123456789012'
 NAMESPACE 'dd8772e1-d792-4fa4-996b-1870577efc0d';
```

   如果您想要對本機資料庫中物件的存取進行更精細的控制，請在建立資料庫時使用 WITH PERSOMMENTS 子句。這可讓您在步驟 4 中為資料庫中的物件授予物件層級權限。

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF ACCOUNT
 '123456789012' NAMESPACE 'dd8772e1-d792-4fa4-996b-1870577efc0d';
```

   您可以透過查詢 SVV_REDSHIFT_DATABASES 檢視來查看您從資料共用中建立的資料庫。您無法連線到從資料共用中建立的這些資料庫，這些資料庫僅限讀取。不過，您可以連線到取用者叢集上的本機資料庫，並執行跨資料庫查詢，從資料共用建立的資料庫中查詢資料。您不能在現有資料共用中建立的資料庫物件頂端建立資料共用。不過，您可以將資料複製到取用者叢集上的個別資料表中，執行所需的任何處理，然後共用已建立的新物件。

3. (選擇性) 建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
 'public';
```

4. 視需要，將從資料共用建立的資料庫和結構描述參照的許可授予使用者叢集中的使用者或角色。如需詳細資訊，請參閱 GRANT 或 REVOKE 。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

如果您在沒有 WITH PERSOMMENCE 的情況下建立資料庫，則只能將從資料共用建立的整個資料庫的權限指派給您的使用者或角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。如果是這樣，您可以建立指向資料共用中特定結構描述的外部結構描述參照，如上一個步驟所述。然後，您可以在結構描述層級上提供精細的許可。您也可以在共用物件之上建立近期繫結檢視，並使用這些檢視來指派精細的許可。也可以考慮讓生產者叢集以所需的精細程度為您建立其他資料共用。您可以依需要，為從資料共用中建立的資料庫建立盡可能多的結構描述參照。

如果您在步驟 2 中使用 WITH PERMISSIONS 建立資料庫，則必須為共用資料庫中的物件指派物件層級權限。只有 USAGE 權限的使用者在獲得額外的物件層級權限之前，不能存取使用 WITH PERMISSIONS 建立的資料庫中的任何物件。

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

5. 查詢資料共用中共用物件的資料。

在取用者資料庫和取用者叢集結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集中的本機物件。若要做到這一點，請使用 JDBC 或 ODBC 驅動程式或 SVV_ALL 和 SVV_REDSHIFT 檢視。

生產者叢集在每個結構描述中可能會有多個資料庫、資料表和檢視的結構描述。取用者端的使用者只能看到透過資料共用提供的物件子集。這些使用者無法從生產者叢集看到所有中繼資料。此方法有助於透過資料共用提供精細的中繼資料安全控制

您可以繼續連線到本機叢集資料庫。但是現在，您也可以使用三部分 database.schema.table 表示法，從資料共用中建立的資料庫和結構描述中讀取。您可以執行跨越任何和所有可見資料庫的查詢。這些資料庫可以是叢集上的本機資料庫，也可以是從資料共用建立的資料庫。取用者叢集無法連線至從資料共用建立的資料庫。

您可以使用完整資格存取資料。如需詳細資訊，請參閱 使用跨資料庫查詢的範例。

```
SELECT * FROM sales_db.public.tickit_sales_redshift;
```

您只能在共用物件上使用 SELECT 陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

除了執行查詢之外，取用者還可以在共用物件上建立檢視。僅支援近期繫結檢視和具體化視觀表。Amazon Redshift 不支援共用資料的一般檢視。取用者建立的檢視可跨越多個本機資料庫或從資料共用建立的資料庫。如需詳細資訊，請參閱 CREATE VIEW。

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

共用資料 AWS 區域

您可以在 AWS 區域中的 Amazon Redshift 叢集之間共用資料以供讀取。透過跨區域資料共用，您可以跨區域共用資料， AWS 區域 而不需要手動複製資料。您不需要將資料卸載到 Amazon S3，也不需要將資料複製到新的 Amazon Redshift 叢集或執行跨區域快照複製。

透過跨區域資料共用，即使叢集位於不同區域 AWS 帳戶，您 AWS 帳戶 也可以在相同叢集或不同叢集之間共用資料。與位於相同 AWS 帳戶 但不同的 Amazon Redshift 叢集共用資料時 AWS 區域，請遵循與在. AWS 帳戶如需詳細資訊，請參閱 共用資料的讀取存取權限 AWS 帳戶。

如果共用資料的叢集不同 AWS 區域， AWS 帳戶 而且您可以遵循與跨資料共用相同的工作流程， AWS 帳戶 並在用戶叢集上包含區域層級關聯。跨區域資料共用支援與. 中的整個 AWS 帳戶、整個或特定叢集命名空間的資料清理關聯。 AWS 區域 AWS 區域如需共用資料的詳細資訊 AWS 帳戶，請參閱共用資料 AWS 帳戶。

使用來自不同區域的資料時，取用者須支付從生產者區域到取用者區域的跨區域資料傳輸費用。

若要使用資料共用，取用者帳戶管理員可以透過下列三種方式之一來與資料共用建立關聯。

- 與整個 AWS 帳戶 跨越其所有關聯 AWS 區域
- 與 AWS 區域 中的特定關聯 AWS 帳戶
- 與特定叢集命名空間的關聯 AWS 區域

當系統管理員選擇整個時 AWS 帳戶，帳戶 AWS 區域 中不同的所有現有和 future 的叢集命名空間都可以存取資料庫。消費者帳戶管理員也可以選擇區域內的特定命名空間 AWS 區域 或叢集命名空間，以授與資料存取權限。

如果您是生產者叢集管理員或資料庫擁有者，請建立資料共用、將資料庫物件和資料取用者新增至資料共用，並將許可授予資料取用者。如需詳細資訊，請參閱 生產者叢集管理員動作。

如果您是生產者帳戶管理員，請使用 AWS Command Line Interface (AWS CLI) 或 Amazon Redshift 主控台授權資料庫，然後選擇資料取用者。

如果您是取用者帳戶管理員，請依照下列步驟執行：

若要將從其他帳戶共用的一或多個資料庫與您的整個 AWS 帳戶 或特定 AWS 區域 或叢集命名空間建立關聯 AWS 區域，請使用 Amazon Redshift 主控台。

透過跨區域資料共用功能，您可以使用 AWS Command Line Interface (AWS CLI) 或 Amazon Redshift 主控台在特 AWS 區域 定叢集中新增叢集。

若要指定一或多個 AWS 區域，您可以使用 `associate-data-share-consumer` CLI 命令搭配選擇性選`consumer-region`項。

使用 CLI 時，下列範例會將`Salesshare`與整個`associate-entire-account`選項 AWS 帳戶 相關聯。您一次只能與一個區域建立關聯。

```
aws redshift associate-data-share-consumer
--region {PRODUCER_REGION}
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--associate-entire-account
```

以下範例會將 `Salesshare` 與美國東部 (俄亥俄) 區域 (`us-east-2`) 產生關聯。

```
aws redshift associate-data-share-consumer
--region {PRODUCER_REGION}
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:0123456789012:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--consumer-region 'us-east-2'
```

下列範例會將`Salesshare`與亞太區域 (雪梨) 區域 () AWS 帳戶 中另一個特定的取用者叢集命名空間產生關聯。`ap-southeast-2`

```
aws redshift associate-data-share-consumer
```

```
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--consumer-arn 'arn:aws:redshift:ap-southeast-2:{CONSUMER_ACCOUNT}:namespace:
{ConsumerImmutableClusterId}'
```

您可以使用 Amazon Redshift 主控台將資料庫與您的整個 AWS 帳戶 或特定 AWS 區域 或叢集命名空間建立關聯。 AWS 區域若要這麼做，請登入 https://console.aws.amazon.com/redshiftv2/。然後，將從其他帳戶共用的一或多個資料共用與您的整個 AWS 帳戶或特定 AWS 區域或 AWS 區域內的特定叢集命名空間建立關聯。如需詳細資訊，請參閱 建立資料共用的關聯。

關聯 AWS 帳戶 或特定叢集命名空間之後，資料庫就可供使用。您也可以隨時變更資料共用關聯。將個別叢集命名空間的關聯變更為時 AWS 帳戶，Amazon Redshift 會以資訊覆寫叢集命名空間。 AWS 帳戶 將關聯從某個叢集命名空間變更 AWS 帳戶 為特定叢集命名空間時，Amazon Redshift 會以叢集命名空間 AWS 帳戶 資訊覆寫資訊。將整個區域和叢集命名空間的關聯變更 AWS 帳戶 為特定 AWS 區域和叢集命名空間時，Amazon Redshift 會以特定區域和叢集命名空間 AWS 帳戶 資訊覆寫資訊。

如果您是取用者叢集管理員，則可以建立參照資料共用的本機資料庫，並視需要，將從資料共用建立的資料庫許可授予取用者叢集中的使用者或角色。您也可以在共用物件上建立檢視，以及建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱 取用者叢集管理員動作。

管理跨區域資料共用的成本控制

使用來自不同區域的資料時，取用者須支付從生產者區域到取用者區域的跨區域資料傳輸費用。不同區域的資料傳輸價格不同。費用是根據每次成功執行查詢所掃描的資料位元組計算。如需 Amazon Redshift 定價的相關資訊，請參閱 Amazon Redshift 定價。

系統會根據位元組數量向您收費 (四捨五入至下一個 MB)，每個查詢最少 10MB。您可以針對查詢使用情況設定成本控制，並檢視叢集上每個查詢傳輸的資料量。

若要監看和控制跨區域資料共用的使用情況和相關成本，您可以建立每日、每週、每月的用量限制，並定義達到這些限制時 Amazon Redshift 自動採取的動作，以透過可預測性協助維持預算。如需 Amazon Redshift 使用量限制的相關資訊，請參閱管理 Amazon Redshift 中的用量限制。

根據您設定的用量限制，Amazon Redshift 採取的動作可以是將事件記錄到系統表、傳送 CloudWatch 警示並使用 Amazon SNS 通知管理員，或是關閉跨區域資料共用以供進一步使用。如需這些動作的相關資訊，請參閱管理 Amazon Redshift 中的用量限制。

若要在 Amazon Redshift 主控台中建立用量限制，請在叢集的動作下選擇設定用量限制。您可以透過「叢集效能」或「監控」索引標籤自動產生的指標，監視使用 CloudWatch 量趨勢，並收到使用量超出

定義限制的警示。或者，您也可以使用 AWS CLI 或 Amazon Redshift API 操作，以程式化方式建立、修改和刪除用量限制。如需詳細資訊，請參閱管理 Amazon Redshift 中的用量限制。

共享授權的 Amazon Redshift 數據 AWS Data Exchange

建立資 AWS Data Exchange 料庫並將其新增至 AWS Data Exchange 產品時，供應商可以在 Amazon Redshift 中授權資料，讓消費者在擁有有效訂閱時，在 Amazon Redshift 中探索、訂閱和查詢 up-to-date 資料。 AWS Data Exchange

將 AWS Data Exchange 資料庫新增至 AWS Data Exchange 產品後，消費者會在訂閱開始時自動存取產品的資料庫，並且只要訂閱處於作用中狀態，即可保留其存取權。

以 AWS Data Exchange 生產者身份使用資料庫

如果您是生產者叢集管理員，請按照下列步驟在 Amazon Redshift 主控台上管理 AWS Data Exchange 資料庫：

1. 在叢集中建立資料庫以共用資料 AWS Data Exchange 並授予資料存取 AWS Data Exchange 權限。

   叢集超級使用者和資料庫擁有者可以建立資料共用。每個資料共用都會在建立期間與資料庫相關聯。只有來自該資料庫的物件才能在該資料共用中共用。可以在具有相同或不同物件細微程度的相同資料庫上建立多個資料共用。可以在叢集中建立的資料共用數量不限。

   您也可以使用 Amazon Redshift 主控台建立資料共用。如需詳細資訊，請參閱 建立資料共用。

   使用管理 ADX 選項，在執行建立資料清單陳述式 AWS Data Exchange 時，隱含地授與資料存取權限。這表示 AWS Data Exchange 管理此資料識別。您只能在建立新資料共用時使用 MANAGEDBY ADX 選項。您不能使用 ALTER DATASHARE 陳述式來修改現有的資料共用，以新增 MANAGEDBY ADX 選項。使用 MANAGEDBY ADX 選項建立資料共用後，只有 AWS Data Exchange 能存取和管理資料共用。

   ```
   CREATE DATASHARE salesshare
   [[SET] MANAGEDBY [=] {ADX} ];
   ```

2. 將物件新增到資料共用。製作者管理員會繼續管理資料清單中可用的資料清單物件。 AWS Data Exchange

   若要將物件新增至資料共用，請在新增物件之前先新增結構描述。當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。您必須明確地新增這些物件。如需詳細資訊，請參閱 ALTER DATASHARE。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

您也可將檢視新增到資料共用。

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM
 public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;
```

使用 ALTER DATASHARE 來共用結構描述及指定結構描述中的資料表、檢視和函數。超級使用者、資料共用擁有者或在資料共用上擁有 ALTER 或 ALL 許可的使用者可以修改資料共用，以在其中新增或移除物件。使用者應具有在資料共用中新增或移除物件的許可。使用者也必須是物件的擁有者，或具有物件的 SELECT、USAGE 或 ALL 許可。

使用 INCLUDENEW 子句將指定結構描述中建立的任何新資料表、檢視或 SQL 使用者定義函數 (UDF) 新增至資料共用。只有超級使用者可以為每個「資料共用-結構描述」配對修改此屬性。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除物件。如需詳細資訊，請參閱將資料共用物件新增到資料共用、從資料共用中移除資料共用物件及編輯 AWS Data Exchange 資料庫。

3. 若要授權存取的資料庫 AWS Data Exchange，請執行下列其中一個動作：

   - AWS Data Exchange 通過在 API 中使用ADX關鍵字明確授權對數據存取權限。`aws redshift authorize-data-share`這允許 AWS Data Exchange 識別服務帳戶中的數據清單並管理將消費者與數據護理相關聯。

     ```
     aws redshift authorize-data-share
     --data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
     {PRODUCER_CLUSTER_NAMESPACE}/salesshare
     --consumer-identifier ADX
     ```

     您可以針對 `AuthorizeDataShare` 和 `DeauthorizeDataShare` API 使用條件金鑰 `ConsumerIdentifier`，明確允許或拒絕 AWS Data Exchange 呼叫 IAM 政策中的兩個 API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Deny",
            "Action": [
                "redshift:AuthorizeDataShare",
                "redshift:DeauthorizeDataShare"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "redshift:ConsumerIdentifier": "ADX"
                }
            }
        }
    ]
}
```

- 使用 Amazon Redshift 主控台來授權或移除 AWS Data Exchange 資料存取的授權。如需詳細資訊，請參閱 授予或移除資料共用的授權。
- 或者，您可以在將資料清單匯入 AWS Data Exchange 資料集時，隱含地授權存取資料清單。AWS Data Exchange

若要移除存取 AWS Data Exchange 資料存取權的授權，請在 API 作業中使用ADX關鍵字。aws redshift deauthorize-data-share透過這樣做，您可以讓 AWS Data Exchange 識別服務帳戶中的資料共用，並管理從資料共用中刪除關聯的操作。

```
aws redshift deauthorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier ADX
```

4. 列出在叢集中建立的資料共用，並查看資料共用的內容。

   下列範例會顯示名為 salesshare 之資料共用的資訊。如需詳細資訊，請參閱 DESC DATASHARE 及 SHOW DATASHARES。

```
DESC DATASHARE salesshare;
```

```
 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name          |   include_new
------------------+--------------------------------------+------------
+------------+-------------+---------------------------------+------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_users_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_venue_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_category_redshift|
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_date_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_event_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_listing_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | table       | public.tickit_sales_redshift   |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | schema      | public                         |  t
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare
 | view        | public.sales_data_summary_view |
```

下列範例顯示生產者叢集上的輸出資料共用。

```
SHOW DATASHARES LIKE 'sales%';
```

輸出結果類似如下。

```
share_name | share_owner  | source_database | consumer_database | share_type |
 createdate      | is_publicaccessible  | share_acl | producer_account |
 producer_namespace
-----------+-------------+-----------------+-------------------+------------
+---------------------+---------------------+-----------+------------------
+--------------------------------------
salesshare |    100       | dev             |                   |  OUTBOUND
 | 2020-12-09 02:27:08 |          True        |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

如需詳細資訊，請參閱 DESC DATASHARE 及 SHOW DATASHARES。

您也可以使用 SVV_DATASHARES、SVV_DATASHARE_CONSUMERS 和 SVV_DATASHARE_OBJECTS 來檢視資料共用、資料共用中的物件，以及資料共用取用者。

5. 捨棄資料共用。我們建議您不要刪除使用 DROP AWS Data Exchange DATASHARE 陳述式共 AWS 帳戶 用給其他人的資料清單。這些帳戶將會失去資料共用的存取權。此動作不可復原。這可能會違反中的資料產品優惠條款 AWS Data Exchange。如果您要刪除 AWS Data Exchange 資料清單，請參閱。DROP DATASHARE 使用須知

   以下範例會捨棄名為 salesshare 的資料共用。

   ```
   DROP DATASHARE salesshare;
   ERROR:  Drop of ADX-managed datashare salesshare requires session variable
    datashare_break_glass_session_var to be set to value '620c871f890c49'
   ```

   若要允許刪除資料識別，請設定 AWS Data Exchange 資料存放區域 _ 玻璃工作階段變數，然後再次執行「刪除資料清除」陳述式。如果您要刪除 AWS Data Exchange 資料清單，請參閱。DROP DATASHARE 使用須知

   您也可以使用 Amazon Redshift 主控台刪除資料共用。如需詳細資訊，請參閱 刪除 AWS Data Exchange 帳戶中建立的資料庫。

6. 使用 ALTER DATASHARE 隨時從資料共用中點移除物件。使用 REVOKE USAGE ON 來撤銷特定取用者在資料共用上的許可。它會撤銷資料共用中物件的 USAGE 許可，並立即停止對所有取用者叢集的存取。存取權被撤銷後，列出資料共用和中繼資料查詢 (例如列出資料庫和資料表) 不會傳回共用物件。

   ```
   ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
   ```

   您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱 編輯 AWS Data Exchange 資料庫。

7. 授予或撤銷來自 AWS Data Exchange 數據庫的 GRANT 使用情況。您不能授予或撤銷 AWS Data Exchange 數據保護的 GRANT 使用情況。下列範例顯示當授與管理的資料存取權限的 GRANT USING 權限時，會顯示錯誤。 AWS 帳戶 AWS Data Exchange

   ```
   CREATE DATASHARE salesshare MANAGEDBY ADX;
   ```

   ```
   GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '012345678910';
   ```

```
ERROR:  Permission denied to add/remove consumer to/from datashare salesshare.
 Datashare consumers are managed by ADX.
```

如需詳細資訊，請參閱 GRANT 或 REVOKE 。

如果您是生產者叢集管理員，請依照下列步驟在主控台上建立並發佈資料清單產品： AWS Data Exchange

- 建立資 AWS Data Exchange 料清單後，生產者會建立新的資料集、匯入資產、建立修訂，以及建立和發行新產品。

  使用 Amazon Redshift 主控台建立資料集。如需詳細資訊，請參閱 建立資料集 AWS Data Exchange。

  如需詳細資訊，請參閱提供資料產品 AWS Data Exchange。

作為消費 AWS Data Exchange 者使用數據庫

如果您是消費者，請依照下列步驟探索包含 AWS Data Exchange 資料存取的資料產品，並查詢 Amazon Redshift 資料：

1. 在 AWS Data Exchange 主控台上，探索並訂閱包含 AWS Data Exchange 資料存取的資料產品。

   訂閱開始後，您可以將授權的 Amazon Redshift 資料存取為資產匯入到包含資料存取的資 AWS Data Exchange 料集。

   有關如何開始使用包含資料 AWS Data Exchange 庫的資料產品的詳細資訊，請參閱上的訂閱資料產品。 AWS Data Exchange
2. 如有需要，在 Amazon Redshift 主控台上，建立 Amazon Redshift 叢集。

   如需如何建立叢集的資訊，請參閱建立叢集。
3. 列出可供您使用的資料共用，並檢視資料共用的內容。如需詳細資訊，請參閱 DESC DATASHARE 及 SHOW DATASHARES。

   下列範例會顯示指定生產者命名空間的輸入資料共用資訊。當您以取用者叢集管理員身分執行 DESC DATASHARE 時，您必須指定 ACCOUNT 和 NAMESPACE 選項以檢視輸入資料共用。

   ```
   DESC DATASHARE salesshare of ACCOUNT '123456789012' NAMESPACE
    '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
   ```

```
 producer_account  |          producer_namespace          | share_type | share_name
 | object_type |           object_name           |   include_new
------------------+-------------------------------------+------------
+------------+-------------+---------------------------------+------------------
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_users_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_venue_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_category_redshift |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_date_redshift     |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_event_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_listing_redshift  |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | table       | public.tickit_sales_redshift    |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | schema      | public                          |
 123456789012      | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare
 | view        | public.sales_data_summary_view  |
```

只有叢集超級使用者才能執行此動作。您也可以使用 SVV_DATASHARES 來檢視資料共用，以及使用 SVV_DATASHARE_OBJECTS 來檢視資料共用內的物件。

下列範例顯示取用者叢集中的輸入資料共用。

```
SHOW DATASHARES LIKE 'sales%';


 share_name | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace
------------+-------------+-----------------+-------------------+------------
+------------+---------------------+-----------+------------------
+--------------------------------------
 salesshare |             |                 |                   | INBOUND
    |            |         t           |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

4. 建立參照資料共用的本機資料庫。您必須指定 [帳戶和命名空間] 選項，才能建立資料庫的本機 AWS Data Exchange 資料庫。如需詳細資訊，請參閱 CREATE DATABASE。

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF ACCOUNT '123456789012'
 NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

如果您想要對本機資料庫中物件的存取進行更精細的控制，請在建立資料庫時使用 WITH PERSOMMENTS 子句。這可讓您在步驟 6 中為資料庫中的物件授予物件層級權限。

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF ACCOUNT
 '123456789012' NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

您可以透過查詢 SVV_REDSHIFT_DATABASES 檢視來查看您從資料共用中建立的資料庫。您無法連線到從資料共用中建立的這些資料庫，這些資料庫僅限讀取。不過，您可以連線到取用者叢集上的本機資料庫，並執行跨資料庫查詢，從資料共用建立的資料庫中查詢資料。您不能在現有資料共用中建立的資料庫物件頂端建立資料共用。不過，您可以將資料複製到取用者叢集上的個別資料表中，執行所需的任何處理，然後共用已建立的新物件。

您也可以使用 Amazon Redshift 主控台從資料共用中建立資料共用。如需詳細資訊，請參閱 從資料共用中建立資料庫。

5. (選擇性) 建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
 'public';
```

6. 視需要，將從資料共用建立的資料庫和結構描述參照的許可授予使用者叢集中的使用者或角色。如需詳細資訊，請參閱 GRANT 或 REVOKE 。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

如果您在沒有 WITH PERSOMMENCE 的情況下建立資料庫，則只能將從資料共用建立的整個資料庫的權限指派給您的使用者和角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。如果是這樣，您可以建立指向資料共用中特定結構描述的外部結構描述參考 (如上一個步驟所述)，並在結構描述層級上提供精細的許可。

您也可以在共用物件之上建立近期繫結檢視，並使用這些檢視來指派精細的許可。也可以考慮讓生產者叢集以所需的精細程度為您建立其他資料共用。您可以依需要，為從資料共用中建立的資料庫建立盡可能多的結構描述參照。

如果您在步驟 4 中使用 WITH PERMISSIONS 建立資料庫，則必須為共用資料庫中的物件指派物件層級權限。只有 USAGE 權限的使用者在獲得額外的物件層級權限之前，不能存取使用 WITH PERMISSIONS 建立的資料庫中的任何物件。

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

7. 查詢資料共用中共用物件的資料。

在取用者資料庫和取用者叢集結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集中的本機物件。若要做到這一點，他們會使用 JDBC 或 ODBC 驅動程式或 SVV_ALL 和 SVV_REDSHIFT 檢視。

生產者叢集在每個結構描述中可能會有多個資料庫、資料表和檢視的結構描述。取用者端的使用者只能看到透過資料共用提供的物件子集。這些使用者無法從生產者叢集看到整個中繼資料。此方法有助於透過資料共用提供精細的中繼資料安全控制

您可以繼續連線到本機叢集資料庫。但是現在，您也可以使用三部分 database.schema.table 表示法，從資料共用中建立的資料庫和結構描述中讀取。您可以執行跨越任何和所有可見資料庫的查詢。這些資料庫可以是叢集上的本機資料庫，也可以是從資料共用建立的資料庫。取用者叢集無法連線至從資料共用建立的資料庫。

您可以使用完整資格存取資料。如需詳細資訊，請參閱 使用跨資料庫查詢的範例。

```
SELECT * FROM sales_db.public.tickit_sales_redshift ORDER BY 1,2 LIMIT 5;

 salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime
---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------
       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |    728.00 |
   109.20 | 2008-02-18 02:36:48
       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |     76.00 |
    11.40 | 2008-06-06 05:00:16
       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |    350.00 |
    52.50 | 2008-06-06 08:26:17
```

```
   4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |    175.00 |
26.25 | 2008-06-09 08:38:52
   5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |    154.00 |
23.10 | 2008-08-31 09:17:02
```

您只能在共用物件上使用 SELECT 陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

除了查詢之外，取用者還可以在共用物件上建立檢視。僅支援近期繫結檢視或具體化視觀表。Amazon Redshift 不支援共用資料的一般檢視。取用者建立的檢視可跨越多個本機資料庫或從資料共用建立的資料庫。如需詳細資訊，請參閱 CREATE VIEW。

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

使用 AWS Lake Formation託管數據庫

共用資料以 AWS Lake Formation 讓您集中定義 Amazon Redshift 資料存取 AWS Lake Formation 權限，並限制使用者對資料捕捉中物件的存取。

以生產者身分使用 Lake Formation 管理的資料共用

身為生產者叢集或工作群組管理員，請依照下列步驟將資料共用至 Lake Formation：

1. 在叢集中建立資料庫並授權 AWS Lake Formation 存取資料庫。

   只有叢集超級使用者和資料庫擁有者可以建立資料共用。每個資料共用都會在建立期間與資料庫相關聯。只有來自該資料庫的物件才能在該資料共用中共用。可以在具有相同或不同物件細微程度的相同資料庫上建立多個資料共用。可以在叢集中建立的資料共用數量不限。

   ```
   CREATE DATASHARE salesshare;
   ```

2. 將物件新增到資料共用。生產者叢集或工作群組管理員會繼續管理可用的資料共用物件。若要將物件新增至資料共用，請在新增物件之前先新增結構描述。當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。您必須明確地新增這些物件。如需詳細資訊，請參閱 ALTER DATASHARE。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

您也可將檢視新增到資料共用。支援的檢視為標準檢視、近期繫結檢視和具體化視觀表。

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM
 public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
```

使用 ALTER DATASHARE 來共用結構描述及指定結構描述中的資料表和檢視。超級使用者、資料共用擁有者或在資料共用上擁有 ALTER 或 ALL 許可的使用者可以修改資料共用，以在其中新增或移除物件。資料庫使用者應是物件的擁有者，或具有物件的 SELECT、USAGE 或 ALL 許可。

使用 INCLUDENEW 子句將指定結構描述中建立的任何新資料表和檢視新增至資料共用。只有超級使用者可以為每個「資料共用-結構描述」配對修改此屬性。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

3. 將資料共用的存取權授予 Lake Formation 管理員帳戶。

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '012345678910' VIA DATA CATALOG;
```

若要撤銷使用權，請使用下列命令。

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '012345678910' VIA DATA CATALOG;
```

4. 使用 `aws redshift authorize-data-share` API 操作授權存取 Lake Formation 的資料共用。這麼做可允許 Lake Formation 識別服務帳戶中的資料共用，並管理將取用者與資料共用的關聯。

```
aws redshift authorize-data-share
```

```
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier {"DataCatalog/<consumer-account-id>"}
```

若要從 Lake Formation 管理的資料共用中移除授權，請使用 `aws redshift deauthorize-data-share` API 操作。這樣，您可 AWS Lake Formation 以識別服務帳戶中的數據照明並刪除授權。

```
aws redshift deauthorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier {"DataCatalog/<consumer-account-id>"}
```

在任何時候，如果生產者叢集或工作群組管理員決定不再需要與取用者叢集或工作群組共用資料，他們可以使用 DROP DATASHARE 刪除資料共用、取消授權資料共用，或撤銷資料共用許可。Lake Formation 中的相關許可和物件不會自動刪除。

```
DROP DATASHARE salesshare;
```

在授權 Lake Formation 帳戶管理資料護理之後，Lake Formation 管理員可以探索共用資料存放區，將 dateshare 與資料目錄 ARN 相關聯，並在連結至資料指向資料指令時建立資料庫。AWS Glue Data Catalog 若要使用關聯資料庫，請使用指 AWS CLI令。 associate-data-share-consumer若要在其中共用資料清單 AWS 區域，請在`associate-data-share-consumer`命令中指定`--region`參數，或使用 AWS 主控台選擇資料取用者。以下範例示範如何跨區域共用 Lake Formation 管理的資料共用。

```
aws redshift associate-data-share-consumer --region <region-1>
--data-share-arn 'arn:aws:redshift:us-
east-1:12345678912:datashare:035c45ea-61ce-86f0-8b75-19ac6102c3b7/sample_share'
--consumer-arn 'arn:aws:glue:<region-1>:111912345678:catalog'
```

Lake Formation 管理員還必須建立本機資源，以定義資料共用中的物件應如何對應到 Lake Formation 中的物件。如需探索資料共用和建立本機資源的相關資訊，請參閱管理 Amazon Redshift 資料共用中的資料許可。

以取用者身分使用 Lake Formation 管理的資料共用

AWS Lake Formation 管理員發現資料查看邀請並在連結至資料準備的資料庫中建立資料庫後，消費者叢集或工作群組管理員可以將叢集與中的資料清單和資料庫建立關聯，建立消費者叢集或工作群組的本機資料庫，並授與 Amazon Redshift 取用者叢集或工作群組中的使用者和角色的存取權以開始查詢。AWS Glue Data Catalog AWS Glue Data Catalog請遵循以下步驟來設定查詢許可。

1. 如有需要，請在 Amazon Redshift 主控台建立 Redshift 叢集做為取用者叢集或工作群組使用。如需如何建立叢集的資訊，請參閱建立叢集。
2. 若要列出使用者叢集或工作群組使用者中有權存取的資料庫，請執行 SHOW DATABASE 命令。AWS Glue Data Catalog

   ```
   SHOW DATABASES FROM DATA CATALOG [ACCOUNT <account-id>,<account-id2>] [LIKE
    <expression>]
   ```

   這樣做會列出資料目錄中可用的資源，例如資料 AWS Glue 庫的 ARN、資料庫名稱以及有關資料清單的資訊。
3. 使用 SHOW 資料 AWS Glue 庫中的資料庫 ARN，在用戶叢集或工作群組中建立本機資料庫。如需詳細資訊，請參閱 CREATE DATABASE。

   ```
   CREATE DATABASE lf_db FROM ARN <lake-formation-database-ARN> WITH [NO] DATA CATALOG
    SCHEMA [<schema>];
   ```
4. 視需要，將從資料共用建立的資料庫和結構描述參照的存取權授予取用者叢集或工作群組中的使用者和角色。如需詳細資訊，請參閱 GRANT 或 REVOKE。請注意，透過 CREATE USER 命令建立的使用者無法存取已共用至 Lake Formation 的資料共用中的物件。只有同時擁有 Redshift 和 Lake Formation 存取權的使用者才能存取已與 Lake Formation 共用的資料共用。

   ```
   GRANT USAGE ON DATABASE sales_db TO IAM:Bob;
   ```

   身為取用者叢集或工作群組管理員，您只能將從資料共用建立的整個資料庫的許可指派給使用者和角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。

   您也可以在共用物件之上建立近期繫結檢視，並使用這些檢視來指派精細的許可。您也可以考慮讓生產者叢集或工作群組以所需的精細程度為您建立其他資料共用。您可以為從資料共用中建立的資料庫建立盡可能多的結構描述參照。
5. 數據庫用戶可以使用視圖 SVV_EXTERNAL_TABLE 和 SVV_EXTERNAL_列來查找數據庫中的所有共享表或列 AWS Glue

```
SELECT * from svv_external_tables WHERE redshift_database_name = 'lf_db';

SELECT * from svv_external_columns WHERE redshift_database_name = 'lf_db';
```

6. 查詢資料共用中共用物件的資料。

   在取用者資料庫和取用者叢集或工作群組結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集或工作群組中的本機物件。若要這麼做，他們可以使用 JDBC 或 ODBC 驅動程式或 SVV_ALL 和 SVV_EXTERNAL 檢視。

   ```
   SELECT * FROM lf_db.schema.table;
   ```

   您只能在共用物件上使用 SELECT 陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

   ```
   // Connect to a local cluster database

   // Create a view on shared objects and access it.

   CREATE VIEW sales_data
   AS SELECT *
   FROM sales_db.public.tickit_sales_redshift
   WITH NO SCHEMA BINDING;

   SELECT * FROM sales_data;
   ```

使用主控台管理資料共用

使用 Amazon Redshift 主控台管理在您帳戶中建立或從其他帳戶共用的資料共用。

您需要建立、編輯或刪除資料共用的許可。如需相關資訊，請參閱在 Amazon Redshift 中管理資料共用的許可。

- 如果您是生產者叢集管理員，則可以建立資料共用、新增資料取用者、新增資料共用物件、從資料共用中建立資料庫、編輯資料共用，或從 CLUSTERS 索引標籤刪除資料共用。

  從瀏覽功能表中，瀏覽叢集索引標籤，從叢集清單中選擇叢集。然後執行下列其中一項：

- 選擇資料共用索引標籤，從在我的命名空間中建立的資料庫中選擇一個資料共用。然後執行下列其中一項：
  - 建立資料共用

    建立資料共用時，您可以新增資料共用物件或資料取用者。如需詳細資訊，請參閱 將資料共用物件新增到資料共用 及 將資料取用者新增至資料共用。
  - 編輯在您帳戶中建立的資料共用
  - 刪除在您帳戶中建立的 資料共用
- 選擇資料共用，然後從來自其他叢集的資料共用區段中選擇資料共用。然後執行下列其中一項：
  - 建立資料共用
  - 從資料共用中建立資料庫
- 選擇資料庫，然後從資料庫區段中選擇資料庫。然後選擇建立資料共用。如需詳細資訊，請參閱從資料共用中建立資料庫。

> **Note**
>
> 若要檢視資料庫和資料庫中的物件，或檢視叢集中的資料共用，請連線至資料庫。如需詳細資訊，請參閱 連線至資料庫。

連線至資料庫

連線至資料庫以檢視此叢集中的資料庫和資料庫中的物件，或檢視資料共用。

用來連線至指定資料庫的使用者憑證必須具備檢視所有資料共用的必要許可。

如果沒有本機連線，請執行下列其中一項動作：

- 在叢集詳細資訊頁面中，從資料庫索引標籤的資料庫或資料共用物件區段中，選擇連線到資料庫以檢視叢集中的資料庫物件。
- 在叢集詳細資訊頁面中，從資料共用索引標籤中執行下列其中一項操作：
  - 在來自其他叢集的資料共用區段中，選擇連線到資料庫以檢視其他叢集的資料共用。
  - 在我的叢集中建立的資料共用區段中，選擇連線到資料庫以檢視您叢集中的資料共用。
- 在連線到資料庫視窗中，執行下列其中一項動作：
  - 如果您選擇建立新連線，請選擇 AWS Secrets Manager 來使用預存密碼來驗證連線的存取權。

或者，選擇暫時憑證以使用資料庫憑證來驗證連線的存取權。指定資料庫名稱和資料庫使用者的值。

選擇連線。

- 選擇使用最近使用的連線來連線到您擁有必要許可的其他資料庫。

  Amazon Redshift 會自動進行連線。

建立資料庫連線之後，您可以開始建立資料共用、查詢資料共用或從資料共用建立資料庫。

建立資料共用

建立資料共用

身為生產者叢集管理員，您可以從叢集詳細資訊頁面的資料庫或資料共用索引標籤中建立資料共用。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 在叢集詳細資訊頁面上，執行下列作業：
   - 從資料庫索引標籤的資料庫區段中，選擇資料庫。資料庫詳細資訊頁面會隨即出現。

     選擇建立資料共用。您只能從本機資料庫建立資料共用。如果您尚未連線到資料庫，則會出現連線到資料庫頁面。依照 連線至資料庫 中的步驟連線到資料庫。如果有最近使用的連線，就會顯示建立資料共用頁面。
   - 如果您沒有資料庫連線，請從資料共用索引標籤的資料共用區段中連線到資料庫。

     在在我的叢集中建立的資料共用區段中，選擇建立資料共用。建立資料共用頁面會隨即出現。
4. 在資料共用資訊區段中，選擇下列其中一項：
   - 選擇資料共用來建立資料共用，以便在不同的 Amazon Redshift 叢集或相同 AWS 帳戶 或不同 AWS 帳戶之間共用資料以供讀取。
   - 選擇 [AWS Data Exchange 資料清單] 以建立資料封存以透過授權您的資料。 AWS Data Exchange
5. 指定資料共用名稱、資料庫名稱和可公開存取的值。

   當您變更資料庫名稱時，請建立新的資料庫連線。

6. 在資料共用物件區段中，選擇新增。新增資料共用頁面會隨即出現。若要將物件新增至資料共用，請遵循 將資料共用物件新增到資料共用。
7. 在「資料取用者」區段中，您可以選擇發佈至 Redshift 帳戶，或發佈至 AWS Glue Data Catalog，這會啟動透過 Lake Formation 共用資料的程序。將您的資料共用發佈到 Redshift 帳戶意味著與另一個作為取用者叢集的 Redshift 帳戶共用您的資料。

> **Note**
>
> 建立資料共用之後，您就無法編輯要發佈到其他選項的組態。

8. 選擇建立資料共用。

Amazon Redshift 會建立資料共用。建立資料共用之後，您就可以從資料共用中建立資料庫。

將資料共用物件新增到資料共用

將一個或多個物件新增到資料共用。對於資料取用者而言，資料共用物件是唯讀的。

您可以在不新增資料共用物件的情況下建立資料共用，稍後再新增物件。

只有當您將至少一個物件新增至資料共用時，資料共用才會變為作用中狀態。

1. 從資料共用清單中選擇要新增物件的資料共用。
2. 選擇新增。新增資料共用物件頁面會隨即出現。
3. 在新增其他資料共用物件之前，請至少向資料共用新增一個結構描述。選擇新增並重複來新增多個結構描述。
4. 您可以選擇從指定結構描述新增所選物件類型的所有現有物件，或從指定的結構描述新增特定個別物件。選擇物件類型，例如資料表和檢視或使用者定義的函數。
5. 您可以選擇新增並重複來新增指定的結構描述和資料共用物件，並繼續新增另一個結構描述和物件。

將資料取用者新增至資料共用

您可以將一個或多個資料取用者新增至資料共用。資料取用者可以是唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

您必須明確選擇關閉或開啟與具有公用存取權的叢集共用您的資料共用。

- 選擇將叢集命名空間新增至資料共用。命名空間是 Amazon Redshift 叢集的全域唯一識別碼 (GUID)。
- 選擇新增 AWS 帳戶 至資料共用。指定的 AWS 帳戶 必須具有資料清單的存取權限。

授予或移除資料共用的授權

身為生產者叢集管理員，請選擇要授權存取資料共用或從中移除授權的資料取用者。獲授權的資料取用者會收到對資料共用採取動作的通知。如果您要將叢集命名空間新增為資料取用者，則不必執行授權。

先決條件：若要授予或移除資料共用的授權，必須至少已將一個資料取用者新增至資料共用。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。
3. 選擇在我的帳戶中。
4. 在我帳戶中的資料共用區段中，執行下列其中一項動作：
   - 選擇您要授權的一個或多個取用者叢集。授權資料取用者頁面會隨即出現。然後選擇授權。

     如果您在建立資料共用時選擇發佈至 AWS Glue Data Catalog，則只能將資料共用的授權授予 Lake Formation 帳戶。

     對於 AWS Data Exchange 資料清理，您一次只能授權一個資料識別。

     當您授權 AWS Data Exchange 資料存取時，即表示您與 AWS Data Exchange 服務共用資料保護，並允許代表您管理 AWS Data Exchange 對資料存取的權限。 AWS Data Exchange 在消費者訂閱產品時，將消費者帳戶作為資料消費者新增至資 AWS Data Exchange 料保護，以允許他們存取。 AWS Data Exchange 沒有對數據保護的讀取權限。
   - 選擇您要從中移除授權的一個或多個取用者叢集。然後選擇移除授權。

授權資料取用者之後，他們就可以存取資料共用物件，並建立取用者資料庫來查詢資料。

移除授權後，資料取用者會立即失去對資料共用的存取權限。

以取用者身分管理來自其他帳戶的資料共用

建立資料共用的關聯

身為消費者叢集系統管理員，您可以將一或多個從其他帳戶共用的資料庫與整個帳 AWS 戶或帳戶中的特定叢集命名空間建立關聯。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇您要建立關聯的資料共用，然後選擇建立關聯。當顯示 [與資料共用建立關聯] 頁面時，選擇下列其中一個關聯類型：
   - 選擇 [整個 AWS 帳戶]，將帳戶中不同 AWS 區域的所有現有和 future 叢集命名空間與 AWS 資料清單產生關聯。然後選擇建立關聯。

     如果將資料變更發佈到 AWS Glue Data Catalog，您只能將資料保留與整個帳戶相關聯。 AWS
   - 選擇特定 AWS 區域和叢集命名空間，將一或多個 AWS 區域和特定叢集命名空間與資料命名空間產生關聯。
     a. 選擇新增區域，將特定區 AWS 域和叢集命名空間新增至資料命名空間。便會顯示「新增 AWS 區域」頁面。
     b. 選擇 AWS 區域。
     c. 執行以下任意一項：
        - 選擇新增所有叢集命名空間，將此區域中所有現有和未來的叢集命名空間新增至資料共用。
        - 選擇新增特定叢集命名空間，將此區域中的一或多個特定叢集命名空間新增至資料共用。
        - 選擇一或多個叢集命名空間，然後選擇新增 AWS 區域。
     d. 選擇建立關聯。

如果您要將資料共用與 Lake Formation 帳戶建立關聯，請前往 Lake Formation 主控台建立資料庫，然後定義資料庫的許可。如需詳細資訊，請參閱開發人員指南中的設定 Amazon Redshift 資料存取權限。 AWS Lake Formation 建立資料 AWS Glue 庫或聯合資料庫之後，您可以使用查詢編輯器 v2 或任何偏好的 SQL 用戶端搭配使用者叢集來查詢資料。如需詳細資訊，請參閱 以取用者身分使用 Lake Formation 管理的資料共用。

與資料建立關聯之後，資料共用就會變為可用。

您也可以隨時變更資料共用關聯。將特定區 AWS 域和叢集命名空間的關聯變更為整個 AWS 帳戶時，Amazon Redshift 會以帳戶資訊覆寫特定區域和叢集命名空間資訊。 AWS 然後， AWS 帳戶中的所有 AWS 區域和叢集命名空間都可以存取資料保護。

將特定叢集命名空間的關聯變更為指定 AWS 區域中的所有叢集命名空間時，此區域中的所有叢集命名空間都可以存取資料清單。

從資料取用者移除資料共用的關聯

身為取用者叢集管理員，您可以從資料取用者移除資料共用的關聯。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇要從資料取用者中移除關聯的資料共用。
5. 在資料取用者段落中，選擇一或多個要從中移除關聯的資料取用者。然後選擇移除關聯。
6. 當移除關聯頁面出現時，請選擇移除關聯。

移除關聯之後，資料取用者將失去對資料共用的存取權。您可以隨時變更資料取用者關聯。

拒絕資料共用

身為取用者叢集管理員，您可以拒絕任何狀態為可用或作用中的資料共用。拒絕資料共用後，取用者叢集使用者將失去資料共用的存取權。如果您呼叫 `DescribeDataSharesForConsumer` API 作業，Amazon Redshift 不會傳回已拒絕的資料共用。如果生產者叢集管理員執行 `DescribeDataSharesForProducer` API 操作，他們將看到資料共用被拒絕。拒絕資料保護之後，生產者叢集管理員可以再次將資料鎖點授權給用戶叢集，而取用者叢集管理員可以選擇將其 AWS 帳戶與資料清單關聯或拒絕。

如果您的 AWS 帳戶與數據照顧有關聯，並且與 Lake Formation 管理的數據存儲器有待關聯，則拒絕由 Lake Formation 管理的數據清理關聯也會拒絕原始數據存儲。若要拒絕特定關聯，生產者叢集管理員可以從指定的資料共用中移除授權。此動作不會影響其他資料共用。

若要拒絕資料清單，請使用 AWS 主控台、API 作業`RejectDataShare`，或`reject-datashare`在. AWS CLI

若要使用控制 AWS 台拒絕資料追蹤：

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇您要拒絕的資料共用。出現拒絕資料共用頁面時，選擇拒絕。

拒絕資料共用之後，您就無法還原變更。Amazon Redshift 會從清單中移除資料共用。若要再次查看資料共用，生產者管理員必須再次對其授權。

管理現有資料共用

檢視資料共用

從 DATASHARES 或 CLUSTERS 索引標籤檢視資料共用。

- 使用 DATASHARES 索引標籤列出您帳戶或其他帳戶中的資料共用。
  - 若要檢視您帳戶中建立的資料共用，請選擇在我的帳戶中，然後選擇您要檢視的資料共用。
  - 若要檢視從其他帳戶共用的資料庫，請選擇來自其他帳戶，然後選擇您要檢視的資料共用。
- 使用 CLUSTERS 索引標籤可列出叢集中或其他叢集中的資料共用。

  連線到資料庫 如需詳細資訊，請參閱 連線至資料庫。

  然後從來自其他叢集的資料共用或在我叢集中建立的資料共用區段中選擇一個資料共用，以檢視其詳細資訊。

從資料共用中移除資料共用物件

您可以使用下列程序，從資料共用中移除一個或多個物件。

從資料共用中移除一個或多個物件

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。

3. 選擇資料共用。
4. 在在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱 連線至資料庫。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 若要對資料共用移除一或多個資料共用物件，請執行下列其中一項操作：
   - 若要從資料共用中移除結構描述，請選擇一或多個結構描述。然後選擇移除。Amazon Redshift 會從資料共用中移除指定的結構描述和指定結構描述的所有物件。
   - 若要從資料共用中移除資料表和檢視，請選擇一或多個資料表和檢視。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有資料表和檢視。
   - 若要從資料共用中移除使用者定義函數，請選擇一或多個使用者定義函數。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有使用者定義函數。

從資料共用中移除資料取用者

您可以從資料共用中移除一或多個資料取用者。資料取用者可以是唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

從叢集命名空間 ID 或帳戶中選擇一或多個資料用 AWS 戶，然後選擇移除。

Amazon Redshift 會從資料共用中移除指定的資料取用者。您會立即失去資料共用的存取權。

編輯在您帳戶中建立的資料共用

使用主控台編輯在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 在在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱 連線至資料庫。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 在資料共用物件或資料取用者區段中進行任何變更。

> **Note**
>
> 如果您選擇將資料清單發佈到 AWS Glue Data Catalog，則無法編輯組態以將資料存放到其他 Amazon Redshift 帳戶。

7. 選擇儲存變更。

Amazon Redshift 會以這些變更來更新您的資料共用。

刪除在您帳戶中建立的 資料共用

使用主控台刪除在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 在在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱 連線至資料庫。
5. 選擇一或多個您要刪除的資料共用，然後選擇刪除。刪除資料共用頁面會隨即出現。

   刪除與 Lake Formation 共用的資料共用不會自動移除 Lake Formation 中的相關許可。若要將其刪除，請前往 Lake Formation 控制台。
6. 輸入刪除以確認刪除指定的資料共用。
7. 選擇刪除。

刪除資料共用之後，資料共用取用者會失去資料共用的存取權。

查詢資料庫

從資料共用中建立資料庫

若要開始查詢資料共用中的資料，請從資料共用中建立資料庫。僅可從指定資料共用中建立一個資料庫。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 在來自其他叢集的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱 連線至資料庫。
5. 選擇您要在其中建立資料庫的資料共用，然後選擇從資料共用建立資料庫。從資料共用建立資料庫頁面會隨即出現。
6. 在資料庫名稱中，指定資料庫名稱。資料庫名稱必須是 1–64 個英數字元 (僅限小寫字母)，且不能是保留字。
7. 選擇建立。

建立資料庫之後，您就可以查詢資料庫中的資料。

管理 AWS Data Exchange 資料庫

建立資料集 AWS Data Exchange

在上建立資料集 AWS Data Exchange。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 在我的帳戶中創建的數據庫部分中，選擇一個數據保護。 AWS Data Exchange
5. 選擇 [建立資料集於] AWS Data Exchange。如需詳細資訊，請參閱發佈新產品。

編輯 AWS Data Exchange 資料庫

使用控制台編輯 AWS Data Exchange 數據庫。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

對於 AWS Data Exchange 資料存取者，您無法變更資料取用者。

若要編輯資 AWS Data Exchange 料存取的可公開存取設定，請使用查詢編輯器 v2。Amazon Redshift 會產生隨機的一次性值來設定工作階段變數，以允許關閉此設定。如需詳細資訊，請參閱 ALTER DATASHARE 使用須知。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 從導覽功能表中選擇編輯器，然後選擇查詢編輯器 v2。
4. 如果這是您第一次使用查詢編輯器 v2，請設定您的 AWS 帳戶. 默認情況下， AWS 擁有的密鑰用於加密資源。如需有關設定的詳細資訊 AWS 帳戶，請參閱 Amazon Redshift 管理指南 AWS 帳戶中的設定。
5. 若要連線到資 AWS Data Exchange 料所在的叢集，請在樹狀檢視面板中選擇 [資料庫] 和 [叢集名稱]。若出現提示，請輸入連線參數。
6. 複製下列 SQL 陳述式。下列範例會變更 salesshare 資料共用的可公開存取設定。

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
```

7. 若要執行複製的 SQL 陳述式，請選擇查詢，然後將複製的 SQL 陳述式貼到查詢區域中。接著選擇執行。

   下列錯誤會隨即出現：

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
ERROR:  Alter of ADX-managed datashare salesshare requires session variable
 datashare_break_glass_session_var to be set to value 'c670ba4db22f4b'
```

   值 'c670ba4db22f4b' 是 Amazon Redshift 在非建議操作發生時隨機產生的一次性值。

8. 將下列範例陳述式複製並貼上查詢區域。接著執行命令。此命SET `datashare_break_glass_session_var`令會套用權限，以允許對 AWS Data Exchange 資料查看執行不建議的作業。

```
SET datashare_break_glass_session_var to 'c670ba4db22f4b';
```

9. 再次執行 ALTER DATASHARE 陳述式。

```
ALTER DATASHARE salesshare;
```

Amazon Redshift 會以這些變更來更新您的資料共用。

刪除 AWS Data Exchange 帳戶中建立的資料庫

使用主控台刪除在您帳戶中建立的 AWS Data Exchange 資料庫。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 從導覽功能表中選擇編輯器，然後選擇查詢編輯器 v2。
4. 如果這是您第一次使用查詢編輯器 v2，請設定您的 AWS 帳戶. 默認情況下， AWS 擁有的密鑰用於加密資源。如需有關設定的詳細資訊 AWS 帳戶，請參閱 Amazon Redshift 管理指南 AWS 帳戶中的設定。
5. 若要連線到資 AWS Data Exchange 料所在的叢集，請在樹狀檢視面板中選擇 [資料庫] 和 [叢集名稱]。若出現提示，請輸入連線參數。
6. 複製下列 SQL 陳述式。下列範例會捨棄 salesshare 資料共用。

```
DROP DATASHARE salesshare
```

7. 若要執行複製的 SQL 陳述式，請選擇查詢，然後將複製的 SQL 陳述式貼到查詢區域中。接著選擇執行。

   下列錯誤會隨即出現：

```
ERROR:  Drop of ADX-managed datashare salesshare requires session variable
 datashare_break_glass_session_var to be set to value '620c871f890c49'
```

   值 '620c871f890c49' 是 Amazon Redshift 在非建議操作發生時隨機產生的一次性值。

8. 將下列範例陳述式複製並貼上查詢區域。接著執行命令。此命令`SET datashare_break_glass_session_var`會套用權限，以允許對 AWS Data Exchange 資料查看執行不建議的作業。

```
SET datashare_break_glass_session_var to '620c871f890c49';
```

9. 再次執行 DROP DATASHARE 陳述式。

```
DROP DATASHARE salesshare;
```

刪除資料共用之後，資料共用取用者會失去資料共用的存取權。

刪除共用資 AWS Data Exchange 料清單可能會違反中的資料產品術語。 AWS Data Exchange

使用 AWS CloudFormation管理資料共用

您可以使用佈建資源的 AWS CloudFormation 堆疊來自動化 AWS 資料共用設定。 CloudFormation 堆疊會在同一 AWS 帳戶中的兩個 Amazon Redshift 叢集之間設定資料共用。因此，您可以在不執行 SQL 陳述式的情況下，啟動資料共用來佈建資源。

堆疊會在您指定的叢集上建立資料共用。資料共用包括資料表和範例唯讀資料。您的其他 Amazon Redshift 叢集可以讀取此資料。

如果您想要透過執行 SQL 陳述式來設定資料清單並授與權限而不使用，開始在 AWS 帳戶中共用 CloudFormation資料，請參閱。共用資料的讀取存取權限 AWS 帳戶

在執行資料共用 CloudFormation 堆疊之前，您必須先使用具有建立 IAM 角色和 Lambda 函數之權限的使用者登入。您還需要在同一個帳戶中使用兩個 Amazon Redshift 叢集。您可以使用其中一個叢集 (生產者)來共用範例資料，使用另一個叢集 (取用者) 來讀取範例資料。這些叢集的主要需求是每個叢集都使用 RA3 節點。如需其他需求，請參閱 在 Amazon Redshift 中使用資料共用時的考量事項。

如需開始設定 Amazon Redshift 叢集的詳細資訊，請參閱 Amazon Redshift 佈建叢集。如需使用自動化設定的詳細資訊 CloudFormation，請參閱什麼是 AWS CloudFormation？

> **Important**
>
> 在啟動 CloudFormation 堆疊之前，請確定同一帳戶中有兩個 Amazon Redshift 叢集，且叢集使用 RA3 節點。確保每個叢集都有一個資料庫和一個超級使用者。如需詳細資訊，請參閱 CREATE DATABASE 及 superuser。

若要啟動您的 CloudFormation 堆疊以進行 Amazon Redshift 資料共用：

1. 按一下啟動 CFN 堆疊，這會將您帶到中的 CloudFormation 服務。 AWS Management Console

   如果出現系統提示，請登入。

   堆疊建立程序會啟動，參考存放在 Amazon S3 中的 CloudFormation 範本檔案。 CloudFormation 模板是 JSON 格式的文本文件，它聲明構成堆棧的 AWS 資源。如需範本的詳細資訊，請參閱瞭解 CloudFormation 範本基礎知識。
2. 選擇下一步以輸入堆疊詳細資料。

3. 在參數底下，為每個叢集輸入下列內容：

   - 您的 Amazon Redshift 叢集名稱，例如 **ra3-consumer-cluster**
   - 您的資料庫名稱，例如 **dev**
   - 資料庫使用者的名稱，例如 **consumeruser**

   我們建議您使用測試叢集，因為堆疊會建立數個資料庫物件。

   選擇下一步。
4. 堆疊選項隨即出現。

   選擇下一步以接受預設設定。
5. 在 [功能] 下，選擇 [我確認 AWS CloudFormation 可能會建立 IAM 資源]。
6. 選擇建立堆疊。

CloudFormation 使用範本建立 Amazon Redshift 堆疊大約需要 10 分鐘，建立名為的資料指令。myproducer_share堆疊會在堆疊詳細資訊中指定的資料庫中建立資料共用。只有來自該資料庫的物件才能共用。

如果建立堆疊時發生錯誤，請執行下列動作：

- 請確定您為每個 Redshift 叢集輸入正確的叢集名稱、資料庫名稱和資料庫使用者名稱。
- 請確定您的叢集具有 RA3 節點。
- 請確定您已使用有權建立 IAM 角色和 Lambda 函數的使用者登入。如需建立 IAM 角色的相關資訊，請參閱建立 IAM 角色。如需 Λ 函數建立政策的相關資訊，請參閱函數開發。

查詢您建立的資料共用

若要使用下列程序，請確定您在每個所述叢集上有執行查詢所需的許可。

查詢您的資料共用：

1. 使用用戶端工具 (例如 Amazon Redshift 查詢編輯器 v2)，Connect 到建立 CloudFormation堆疊時輸入的資料庫上的生產者叢集。
2. 查詢資料庫。

```
SHOW DATASHARES;
```

```
+-----------------+------------+----------------+------------------
+-----------+-----------+--------------------+----------+-----------------
+-------------------------------------+
|    share_name    | share_owner | source_database | consumer_database | share_type
 | createdate | is_publicaccessible | share_acl | producer_account |
 producer_namespace          |
+-----------------+------------+----------------+------------------
+-----------+-----------+--------------------+----------+-----------------
+-------------------------------------+
| myproducer_share | 100         | sample_data_dev | myconsumer_db     | INBOUND
  | NULL       | true               | NULL      |  producer-acct   |        your-
producer-namespace       |
+-----------------+------------+----------------+------------------
+-----------+-----------+--------------------+----------+-----------------
+-------------------------------------+
```

上述命令會傳回由堆疊建立的資料共用名稱，稱為 myproducer_share。也會傳回與資料共用相關聯的資料庫名稱 (myconsumer_db)。

複製生產者命名空間識別碼，以便在稍後步驟中使用。

3. 描述資料共用中的物件。

```
DESC DATASHARE myproducer_share;

+-----------------+-------------------------------------+-----------
+-----------------+------------+-------------------------------------
+-------------+
| producer_account |          producer_namespace          | share_type |
 share_name    | object_type |             object_name             | include_new |
+-----------------+-------------------------------------+-----------
+-----------------+------------+-------------------------------------
+-------------+
|   producer-acct  |        your-producer-namespace       | OUTBOUND   |
 myproducer_share | schema      | myproducer_schema                   | true
 |
|   producer-acct  |        your-producer-namespace       | OUTBOUND   |
 myproducer_share | table       | myproducer_schema.tickit_sales      | NULL
 |
|   producer-acct  |        your-producer-namespace       | OUTBOUND   |
 myproducer_share | view        | myproducer_schema.ticket_sales_view | NULL
 |
```

```
+-----------------+---------------------------------------+-----------
+-----------------+-------------+-------------------------------------
+-------------+
```

當您描述資料共用時，其會傳回資料表和檢視的屬性。堆疊會將含有範例資料的資料表和檢視新增至生產者資料庫，例如 tickit_sales 和 tickit_sales_view。如需 TICKIT 範例庫的相關資訊，請參閱 範本資料庫。

您不必委派對資料共用執行查詢的許可。堆疊會授予必要許可。

4. 使用用戶端工具連線至取用者叢集。描述資料共用，指定生產者的命名空間。

```
DESC DATASHARE myproducer_share OF NAMESPACE '<namespace id>'; --specify the unique
 identifier for the producer namespace

+-----------------+---------------------------------------+-----------
+-----------------+-------------+-------------------------------------
+-------------+
| producer_account |          producer_namespace          | share_type |
 share_name    | object_type |             object_name             | include_new |
+-----------------+---------------------------------------+-----------
+-----------------+-------------+-------------------------------------
+-------------+
|   producer-acct  |        your-producer-namespace       | INBOUND    |
 myproducer_share | schema      | myproducer_schema                   | NULL
 |
|   producer-acct  |        your-producer-namespace       | INBOUND    |
 myproducer_share | table       | myproducer_schema.tickit_sales      | NULL
 |
|   producer-acct  |        your-producer-namespace       | INBOUND    |
 myproducer_share | view        | myproducer_schema.ticket_sales_view | NULL
 |
+-----------------+---------------------------------------+-----------
+-----------------+-------------+-------------------------------------
+-------------+
```

5. 您可以透過指定資料共用的資料庫和結構描述來查詢資料共用中的資料表。如需詳細資訊，請參閱使用跨資料庫查詢的範例。以下查詢會從 TICKIT 範例資料庫中的 SALES 資料表傳回銷售和賣家資料。如需詳細資訊，請參閱 SALES 資料表。

```
SELECT * FROM myconsumer_db.myproducer_schema.tickit_sales_view;
```

```
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |       728 |
     109.2 | 2008-02-18 02:36:48 |
|       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |        76 |
      11.4 | 2008-06-06 05:00:16 |
|       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |       350 |
      52.5 | 2008-06-06 08:26:17 |
|       4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |       175 |
     26.25 | 2008-06-09 08:38:52 |
|       5 |      6 |    47402 |   14115 |    8240 |   2069 |       2 |       154 |
      23.1 | 2008-08-31 09:17:02 |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

> **Note**
>
> 查詢會針對共用結構描述中的檢視執行。您不能直接連線到從資料共用中建立的資料庫。它們是唯讀金鑰。

6. 若要執行包含彙總的查詢，請使用下列範例。

```
SELECT * FROM myconsumer_db.myproducer_schema.tickit_sales ORDER BY 1,2 LIMIT 5;

+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|       1 |      1 |    36861 |   21191 |    7872 |   1875 |       4 |       728 |
     109.2 | 2008-02-18 02:36:48 |
|       2 |      4 |     8117 |   11498 |    4337 |   1983 |       2 |        76 |
      11.4 | 2008-06-06 05:00:16 |
|       3 |      5 |     1616 |   17433 |    8647 |   1983 |       2 |       350 |
      52.5 | 2008-06-06 08:26:17 |
|       4 |      5 |     1616 |   19715 |    8647 |   1986 |       1 |       175 |
     26.25 | 2008-06-09 08:38:52 |
```

```
|       5 |       6 |    47402 |   14115 |    8240 |   2069 |       2 |       154 |
      23.1 | 2008-08-31 09:17:02 |
+---------+---------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

該查詢會從範例 TICKIT 資料傳回銷售和賣方資料。

如需更多資料共用查詢範例，請參閱 共用資料的讀取存取權限 AWS 帳戶。

使用主控台透過寫入功能管理資料共用 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如需有關設定 PREVIEW_2023 音軌的詳細資訊，請參閱下列其中一項：

- 對於 Amazon Redshift Serverless 預覽：建立預覽工作群組
- 對於 Amazon Redshift 佈建的叢集預覽：建立預覽叢集

如需有關開始使用資料共用的詳細資訊，請移至共用資料的寫入存取權限 (預覽)。

使用 Amazon Redshift 主控台管理在您帳戶中建立或從其他帳戶共用的資料共用。

連線至資料庫 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如需有關開始使用資料共用的詳細資訊，請移至共用資料的寫入存取權限 (預覽)。

連線至資料庫以檢視此叢集中的資料庫和資料庫中的物件，或檢視資料共用。

用來連線至指定資料庫的使用者憑證必須具備檢視所有資料共用的必要許可。

如果沒有本機連線，請執行下列其中一項動作：

- 在叢集詳細資訊頁面中，從資料庫索引標籤的資料庫或資料共用物件區段中，選擇連線到資料庫以檢視叢集中的資料庫物件。
- 在叢集詳細資訊頁面中，從資料共用索引標籤中執行下列其中一項操作：
  - 在來自其他叢集的資料共用區段中，選擇連線到資料庫以檢視其他叢集的資料共用。
  - 在我的叢集中建立的資料共用區段中，選擇連線到資料庫以檢視您叢集中的資料共用。
- 在連線到資料庫視窗中，執行下列其中一項動作：
  - 如果您選擇建立新連線，請選擇 AWS Secrets Manager 來使用預存密碼來驗證連線的存取權。

    或者，選擇暫時憑證以使用資料庫憑證來驗證連線的存取權。指定資料庫名稱和資料庫使用者的值。

    選擇連線。
  - 選擇使用最近使用的連線來連線到您擁有必要許可的其他資料庫。

    Amazon Redshift 會自動進行連線。

建立資料庫連線之後，您可以開始建立資料共用、查詢資料共用或從資料共用建立資料庫。

建立資料共用和新增物件 (預覽)

建立資料共用

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如需有關開始使用資料共用的詳細資訊，請移至共用資料的寫入存取權限 (預覽)。

身為生產者叢集管理員，您可以從叢集詳細資訊頁面的資料庫或資料共用索引標籤中建立資料共用。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。

3. 在叢集詳細資訊頁面上，執行下列作業：

   - 從資料庫索引標籤的資料庫區段中，選擇資料庫。資料庫詳細資訊頁面會隨即出現。

     選擇建立資料共用。您只能從本機資料庫建立資料共用。如果您尚未連線到資料庫，則會出現連線到資料庫頁面。依照 連線至資料庫 (預覽) 中的步驟連線到資料庫。如果有最近使用的連線，就會顯示建立資料共用頁面。

   - 如果您沒有資料庫連線，請從資料共用索引標籤的資料共用區段中連線到資料庫。

     在在我的叢集中建立的資料共用區段中，選擇建立資料共用。隨即出現建立資料共用頁面。

4. 從這裡，您可以新增各種類型的資料庫物件。選擇新增按鈕以新增物件。隨即會出現對話方塊。執行以下步驟：

   1. 選擇一個結構描述或多個結構描述。這樣做可讓結構描述中的物件加入。
   2. 從結構描述選取物件類型。

      從這裡您可以選擇兩個選項來新增物件：

      - 從結構描述新增特定物件 — 如果您選擇此項，它會依名稱列示個別物件。您可以選取物件，並將其新增至資料共用。例如，您可以視需要新增特定的資料表和預存程序。然後，您選取的結構描述中的資料表和預存程序會包含在資料共用中。設定權限會在後續步驟中進一步說明。繼續執行檢視和其他類型，選取要加入的物件。
      - 將所選物件類型中的所有現有物件新增至結構描述 – 這會新增所有物件。

   3. 您也可以選擇是否要新增未來的物件。當您選擇包括新增至結構描述的資料共用物件時，表示新增至結構描述的物件會自動新增至資料共用。
   4. 選擇新增以完成區段，並新增物件。這些全列在資料共用物件之下。
   5. 加入物件之後，您可以選取個別物件，並編輯其權限。如果您選取結構描述，會出現對話方塊，詢問您是否要新增限定範圍權限。這使得結構描述中的每個現有或新增的物件都有一組預先選取的權限，適用於該物件類型。例如，管理員可以設定讓所有新增的資料表都具有 SELECT 和 UPDATE 權限。
   6. 設定結構描述權限後，您可以逐步瀏覽其他物件類型並選取其權限。例如，您可以將 UPDATE 權限新增至特定資料表。
   7. 在「資料取用者」段落中，您可以新增命名空間或新增 AWS 帳戶做為資料護理的用戶。
   8. 選擇建立資料共用以儲存變更。

   建立資料共用之後，它會出現在我的命名空間中建立的資料共用下的清單中。如果您從清單中選擇資料共用，您可以檢視其取用者、物件及其他屬性。

將資料取用者新增至資料共用

您可以將一個或多個資料取用者新增至資料共用。資料取用者可以是唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

您必須明確選擇關閉或開啟與具有公用存取權的叢集共用您的資料共用。

- 選擇將叢集命名空間新增至資料共用。命名空間是 Amazon Redshift 叢集的全域唯一識別碼 (GUID)。
- 選擇新增 AWS 帳戶 至資料共用。指定的 AWS 帳戶 必須具有資料清單的存取權限。

授予或移除資料共用的授權 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如需有關開始使用資料共用的詳細資訊，請移至共用資料的寫入存取權限 (預覽)。

身為生產者叢集管理員，請選擇要授權存取資料共用或從中移除授權的資料取用者。獲授權的資料取用者會收到對資料共用採取動作的通知。如果您要將叢集命名空間新增為資料取用者，則不必執行授權。

先決條件：若要授予或移除資料共用的授權，必須至少已將一個資料取用者新增至資料共用。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。從這裡可以看到名為 資料共用取用者的清單。選擇您要授權的一個或多個取用者叢集。然後選擇授權。
3. 系統顯示授權帳戶對話方塊。您可以選擇一兩種授權類型。
   - 在 [叢集名稱或工作群組名稱] 上唯讀 — 這表示取用者沒有可用的寫入權限，即使資料共用建立者授予寫入權限也一樣。
   - 在 [叢集名稱或工作群組名稱] 上讀取和寫入 — 這表示建立者授予的所有權限 (包括寫入權限) 均可供取用者使用。
4. 選擇儲存。

您也可以授權 AWS Data Exchange 為消費者。

1. 如果您在建立資料共用時選擇發佈至 AWS Glue Data Catalog，則只能將資料共用的授權授予 Lake Formation 帳戶。

   對於 AWS Data Exchange 資料清理，您一次只能授權一個資料識別。

   當您授權 AWS Data Exchange 資料存取時，即表示您與 AWS Data Exchange 服務共用資料保護，並允許代表您管理 AWS Data Exchange 對資料存取的權限。 AWS Data Exchange 在消費者訂閱產品時，將消費者帳戶作為資料消費者新增至資 AWS Data Exchange 料保護，以允許他們存取。 AWS Data Exchange 沒有對數據保護的讀取權限。
2. 選擇儲存。

授權資料取用者之後，他們就可以存取資料共用物件，並建立取用者資料庫來查詢資料。

移除授權：

選擇您要從中移除授權的一個或多個取用者叢集。然後選擇移除授權。

移除授權後，資料取用者會立即失去對資料共用的存取權限。

以取用者身分關聯或拒絕資料共用 (預覽)

建立資料共用的關聯

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如需有關開始使用資料共用的詳細資訊，請移至共用資料的寫入存取權限 (預覽)。

身為消費者叢集系統管理員，您可以將一或多個從其他帳戶共用的資料庫與整個帳 AWS 戶或帳戶中的特定叢集命名空間建立關聯。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。選擇來自其他帳戶。

3. 在來自其他帳戶的資料共用區段中，選擇您要建立關聯的資料共用，然後選擇建立關聯。當顯示關聯資料共用頁面時，選擇下列其中一個關聯類型：

   - 選擇 [整個 AWS 帳戶]，將帳戶中不同 AWS 區域的所有現有和 future 叢集命名空間與 AWS 資料清單產生關聯。

     如果將資料變更發佈到 AWS Glue Data Catalog，您只能將資料保留與整個帳戶相關聯。 AWS
4. 您可以從這裡選擇允許的權限。選擇如下：

   - 唯讀 — 如果您選擇唯讀，則取用者無法使用諸如 UPDATE 或 INSERT 之類的寫入權限，即使這些權限已在生產者上授予與並已授權。
   - 讀取和寫入 — 取用者資料共用使用者將擁有生產者授予和授權的所有權限，包括讀取和寫入。
5. 或選擇 [特定 AWS 區域] 和 [叢集命名空間]，將一或多個 AWS 區域和特定叢集命名空間與資料命名空間產生關聯。選擇新增區域，將特定區 AWS 域和叢集命名空間新增至資料命名空間。便會顯示「新增 AWS 區域」頁面。
6. 選擇 AWS 區域。
7. 執行以下任意一項：

   - 選擇新增所有叢集命名空間，將此區域中所有現有和未來的叢集命名空間新增至資料共用。
   - 選擇新增特定叢集命名空間，將此區域中的一或多個特定叢集命名空間新增至資料共用。
   - 選擇一或多個叢集命名空間，然後選擇新增 AWS 區域。
8. 選擇關聯。

生產者可以傳回並變更授權的設定，這可能會影響取用者的關聯設定。

如果您要將資料共用與 Lake Formation 帳戶建立關聯，請前往 Lake Formation 主控台建立資料庫，然後定義資料庫的許可。如需詳細資訊，請參閱開發人員指南中的設定 Amazon Redshift 資料存取權限。 AWS Lake Formation 建立資料 AWS Glue 庫或聯合資料庫之後，您可以使用查詢編輯器 v2 或任何偏好的 SQL 用戶端搭配使用者叢集來查詢資料。

與資料建立關聯之後，資料共用就會變為可用。

您也可以隨時變更資料共用關聯。將特定區 AWS 域和叢集命名空間的關聯變更為整個 AWS 帳戶時，Amazon Redshift 會以帳戶資訊覆寫特定區域和叢集命名空間資訊。 AWS 然後， AWS 帳戶中的所有 AWS 區域和叢集命名空間都可以存取資料保護。

將特定叢集命名空間的關聯變更為指定 AWS 區域中的所有叢集命名空間時，此區域中的所有叢集命名空間都可以存取資料清單。

從資料取用者移除資料共用的關聯

身為取用者叢集管理員，您可以從資料取用者移除資料共用的關聯。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇要從資料取用者中移除關聯的資料共用。
5. 在資料取用者段落中，選擇一或多個要從中移除關聯的資料取用者。然後選擇移除關聯。
6. 當移除關聯頁面出現時，請選擇移除關聯。

移除關聯之後，資料取用者將失去對資料共用的存取權。您可以隨時變更資料取用者關聯。

拒絕資料共用

身為取用者叢集管理員，您可以拒絕任何狀態為可用或作用中的資料共用。拒絕資料共用後，取用者叢集使用者將失去資料共用的存取權。如果您呼叫 `DescribeDataSharesForConsumer` API 作業，Amazon Redshift 不會傳回已拒絕的資料共用。如果生產者叢集管理員執行 `DescribeDataSharesForProducer` API 操作，他們將看到資料共用被拒絕。拒絕資料保護之後，生產者叢集管理員可以再次將資料鎖點授權給用戶叢集，而取用者叢集管理員可以選擇將其 AWS 帳戶與資料清單關聯或拒絕。

如果您的 AWS 帳戶與數據照顧有關聯，並且與 Lake Formation 管理的數據存儲器有待關聯，則拒絕由 Lake Formation 管理的數據清理關聯也會拒絕原始數據存儲。若要拒絕特定關聯，生產者叢集管理員可以從指定的資料共用中移除授權。此動作不會影響其他資料共用。

若要拒絕資料清單，請使用 AWS 主控台、API 作業`RejectDataShare`，或`reject-datashare`在. AWS CLI

若要使用控制 AWS 台拒絕資料追蹤：

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表中，選擇資料共用。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇您要拒絕的資料共用。出現拒絕資料共用頁面時，選擇拒絕。

拒絕資料共用之後，您就無法還原變更。Amazon Redshift 會從清單中移除資料共用。若要再次查看資料共用，生產者管理員必須再次對其授權。

管理現有資料共用 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如需有關開始使用資料共用的詳細資訊，請移至共用資料的寫入存取權限 (預覽)。

檢視資料共用

從 DATASHARES 或 CLUSTERS 索引標籤檢視資料共用。

- 使用 DATASHARES 索引標籤列出您帳戶或其他帳戶中的資料共用。
  - 若要檢視您帳戶中建立的資料共用，請選擇在我的帳戶中，然後選擇您要檢視的資料共用。
  - 若要檢視從其他帳戶共用的資料庫，請選擇來自其他帳戶，然後選擇您要檢視的資料共用。
- 使用 CLUSTERS 索引標籤可列出叢集中或其他叢集中的資料共用。

  連線到資料庫 如需詳細資訊，請參閱 連線至資料庫 (預覽)。

  然後從來自其他叢集的資料共用或在我叢集中建立的資料共用區段中選擇一個資料共用，以檢視其詳細資訊。

從資料共用中移除資料共用物件

您可以使用下列程序，從資料共用中移除一個或多個物件。

從資料共用中移除一個或多個物件

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。

4. 在在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱 連線至資料庫 (預覽)。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 若要對資料共用移除一或多個資料共用物件，請執行下列其中一項操作：
   - 若要從資料共用中移除結構描述，請選擇一或多個結構描述。然後選擇移除。Amazon Redshift 會從資料共用中移除指定的結構描述和指定結構描述的所有物件。
   - 若要從資料共用中移除資料表和檢視，請選擇一或多個資料表和檢視。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有資料表和檢視。
   - 若要從資料共用中移除使用者定義函數，請選擇一或多個使用者定義函數。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有使用者定義函數。

從資料共用中移除資料取用者

您可以從資料共用中移除一或多個資料取用者。資料取用者可以是唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

從叢集命名空間 ID 或帳戶中選擇一或多個資料用 AWS 戶，然後選擇移除。

Amazon Redshift 會從資料共用中移除指定的資料取用者。您會立即失去資料共用的存取權。

編輯在您帳戶中建立的資料共用

使用主控台編輯在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 於在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 在資料共用物件或資料取用者區段中進行任何變更。

> **Note**
>
> 如果您選擇將資料清單發佈到 AWS Glue Data Catalog，則無法編輯組態以將資料存放到其他 Amazon Redshift 帳戶。

7. 選擇儲存變更。

Amazon Redshift 會以這些變更來更新您的資料共用。

刪除在您帳戶中建立的 資料共用

使用主控台刪除在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 於在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。
5. 選擇一或多個您要刪除的資料共用，然後選擇刪除。刪除資料共用頁面會隨即出現。

   刪除與 Lake Formation 共用的資料共用不會自動移除 Lake Formation 中的相關許可。若要將其刪除，請前往 Lake Formation 控制台。
6. 輸入刪除以確認刪除指定的資料共用。
7. 選擇刪除。

刪除資料共用之後，資料共用取用者會失去資料共用的存取權。

查詢資料共用 (預覽)

這是針對 Amazon Redshift 的多資料倉儲透過資料共用功能進行寫入的發行前版本文件，可在 PREVIEW_2023 追蹤公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版服務參與。

如需有關開始使用資料共用的詳細資訊，請移至共用資料的寫入存取權限 (預覽)。

從資料共用中建立資料庫

若要開始查詢資料共用中的資料，請從資料共用中建立資料庫。僅可從指定資料共用中建立一個資料庫。

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 在來自其他叢集的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱 連線至資料庫 (預覽)。
5. 選擇您要在其中建立資料庫的資料共用，然後選擇從資料共用建立資料庫。從資料共用建立資料庫頁面會隨即出現。
6. 在資料庫名稱中，指定資料庫名稱。資料庫名稱必須是 1–64 個英數字元 (僅限小寫字母)，且不能是保留字。
7. 選擇建立。

建立資料庫之後，您可以查詢資料庫中的資料或執行寫入作業 (如果取用者管理員已授予、授權和關聯)。

在 Amazon Redshift 中擷取和查詢半結構化資料

透過使用 Amazon Redshift 中的半結構化資料支援，您可以在 Amazon Redshift 資料倉儲中擷取和儲存半結構化資料。使用 SUPER 資料類型和 PartiQL 語言，Amazon Redshift 擴展了資料倉儲功能，以與 SQL 和 NoSQL 資料來源整合。如此一來，Amazon Redshift 就能對 JSON 等關聯式和半結構化儲存資料進行有效的分析。

Amazon Redshift 提供兩種形式的半結構化資料支援： SUPER 資料類型和 Amazon Redshift Spectrum。

如果您需要以低延遲插入或更新小批次的 JSON 資料，請使用 SUPER 資料類型。此外，當您的查詢需要強大的一致性、可預測的查詢效能、複雜的查詢支援，以及不斷演變的結構描述和無結構描述資料的易用性時，請使用 SUPER。

相反地，如果您的資料查詢需要與其他 AWS 服務整合，並且主要為了存檔目的而存放在 Amazon S3 中的資料，請將 Amazon Redshift Spectrum 與開放檔案格式搭配使用。

SUPER 資料類型的使用案例

在 Amazon Redshift 中使用 SUPER 資料類型的半結構化資料支援，可提供卓越的效能、彈性和易用性。下列使用案例有助於示範如何將半結構化資料支援與 SUPER 結合使用。

快速靈活地插入 JSON 資料 - Amazon Redshift 支援快速交易，這些交易可以剖析 JSON 並將其儲存為 SUPER 值。與將 SUPER 屬性分解為傳統欄的表格中執行相同的插入操作相比，插入交易的運作速度最多可快五倍。例如，假設傳入的 JSON 格式為 {“a”:.., “b”:.., “c”“..., ...}。透過將傳入的 JSON 儲存到具有單一 SUPER 欄 S 的表格 TJ 中，而不是將其儲存到具有欄 ”a’、‘b’、“c’ 等的傳統表格 TR 中，可以多次加速插入效能。當 JSON 中有數百個屬性時，SUPER 資料類型的效能優勢就會變得很大。

此外，SUPER 資料類型不需要規則結構描述。在儲存傳入的 JSON 之前，您不需要自我檢查和清理傳入的 JSON。例如，假設傳入的 JSON 具有字串 "c" 屬性，而其他 JSON 具有整數 “c” 屬性，但沒有 SUPER 資料類型。在這種情況下，您必須分隔 c_string 和 c_int 欄或清理資料。相反地，對於 SUPER 資料類型，擷取期間會儲存所有 JSON 資料，而不會遺失資訊。稍後，您可以使用 SQL 的 PartiQL 擴充功能來分析資訊。

靈活的探索查詢 - 將半結構化資料 (例如 JSON) 儲存到 SUPER 資料值之後，您可以在不強加結構描述的情況下進行查詢。您可以使用 PartiQL 動態類型和寬鬆的語義來執行查詢並探索您需要的深度巢狀資料，而不需要在查詢之前強加結構描述。

將擷取、載入、轉換 (ETL) 操作的彈性查詢轉換為傳統具體化視觀表 - 將無結構描述和半結構化資料儲存到 SUPER 之後，您可以使用 PartiQL 具體化視觀表來自我檢查資料並將其分解為具體化視觀表。

具有分解資料的具體化視觀表是傳統分析案例的效能和可用性優勢的一個很好的範例。當您對分解資料執行分析時，Amazon Redshift 具體化視觀表的單欄式組織可提供更好的效能。此外，需要擷取資料的傳統結構描述的使用者和商業智慧 (BI) 工具可以使用檢視 (無論是具體化或虛擬化) 做為傳統的資料結構描述呈現方式。

在 PartiQL 具體化視觀表將 JSON 或 SUPER 中的資料擷取到傳統的資料欄式具體化視觀表之後，您可以查詢具體化視觀表。如需 SUPER 資料類型如何與具體化視觀表搭配運作的相關資訊，請參閱使用 SUPER 資料類型搭配具體化視觀表。

您可以將動態資料遮罩政策套用至 SUPER 類型資料欄路徑上的 `scalar` 值。如需動態資料遮罩的詳細資訊，請參閱 動態資料遮罩。如需有關將動態資料遮罩與 SUPER 資料類型搭配使用的詳細資訊，請參閱 搭配 SUPER 資料類型路徑使用動態資料遮罩。(預覽)

如需使用 SUPER 資料類型的相關資訊，請參閱SUPER 類型。

如需使用 SUPER 資料類型的範例，請參閱本主題中開頭為 SUPER 範例資料集 的小節。

SUPER 資料類型使用的概念

接下來，您可以找到一些 Amazon Redshift SUPER 資料類型概念。

了解 Amazon Redshift 中的 SUPER 資料類型是什麼 - SUPER 資料類型是一種 Amazon Redshift 資料類型，可儲存無結構描述陣列和包含 Amazon Redshift 純量以及可能巢狀陣列和結構的結構。SUPER 資料類型可以原生儲存不同格式的半結構化資料，例如 JSON 或來自文件導向來源的資料。您可以新增 SUPER 欄來儲存半結構化資料，並撰寫存取 SUPER 欄的查詢，以及一般的純量欄。如需 SUPER 資料類型的相關資訊，請參閱 SUPER 類型。

將無結構描述 JSON 擷取到 SUPER - 透過靈活的半結構化 SUPER 資料類型，Amazon Redshift 可以接收無結構描述 JSON 並將其擷取到 SUPER 值。例如，Amazon Redshift 可以將 JSON 值 [10.5, "first"] 擷取到 SUPER 值 [10.5, ‘first’]，這是一個包含 Amazon Redshift 十進位 10.5 和 varchar ‘first’ 的陣列。Amazon Redshift 可以使用 COPY 命令或 JSON 剖析函數，將 JSON 擷取到 SUPER 值中，例如 json_parse('[10.5, "first"]')。預設情況下，COPY 和 `json_parse` 都使用嚴格的剖析語義來擷取 JSON。您還可以使用資料庫資料本身來建構 SUPER 值，包括陣列和結構。

在擷取無結構描述 JSON 的不規則結構時，SUPER 欄不需要修改結構描述。例如，在分析點擊流時，您最初將具有屬性「IP」和「時間」的「點擊」結構儲存在 SUPER 欄中。您可以在不變更結構描述的情況下新增屬性「customer id」，以擷取此類變更。

SUPER 資料類型所使用的原生格式是二進位格式，與文字形式的 JSON 值相比，它所需的空間較少。如此可在查詢時加快 SUPER 值的擷取和執行期處理速度。

使用 PartiQL 查詢超級資料 — PartiQL 是目前許多服務目前使用的 SQL-92 的向後相容延伸功能。AWS 透過使用 PartiQL，熟悉的 SQL 建構模組可以順暢地結合對傳統表格式 SQL 資料和 SUPER 的半結構化資料的存取。您可以執行物件和陣列導覽以及解除巢狀化陣列。PartiQL 擴展了標準 SQL 語言，以宣告方式表達和處理巢狀和多值資料。

PartiQL 是 SQL 的擴充功能，其中 SUPER 欄的巢狀和無結構描述資料是一流的公民。PartiQL 不要求在查詢編譯時對所有查詢運算式進行類型檢查。這種方法可讓包含 SUPER 資料類型的查詢運算式，在存取 SUPER 欄內的實際資料類型時，在查詢執行期間以動態方式輸入。此外，PartiQL 在寬鬆模式下運作，其中類型不一致不會導致失敗，但會傳回 null。無結構描述和寬鬆查詢處理的組合使得 PartiQL 非常適合擷取、載入、轉換 (ELT) 應用程式，您的 SQL 查詢會評估擷取至 SUPER 欄的 JSON 資料。

與 Redshift Spectrum 整合 - 當透過 JSON、Parquet 和其他具有巢狀資料的格式執行 Redshift Spectrum 查詢時，Amazon Redshift 支援 PartiQL 的多個面向。Redshift Spectrum 僅支援具有結構描述的巢狀資料。例如，使用 Redshift Spectrum，您可以宣告 JSON 資料在結構描述 ARRAY<STRUCT<a:INTEGER, b:DECIMAL(5,2)>> 中具有屬性 nested_schemaful_example。此屬性的結構描述決定了資料永遠包含陣列，其中包含整數 a 和小數 b 的結構。如果資料變更為包含更多屬性，類型也會變更。相反地，SUPER 資料類型不需要結構描述。您可以使用具有不同屬性或類型的結構元素來儲存陣列。此外，一些值可以儲存在陣列之外。

如需支援 SUPER 資料類型之函數的詳細資訊，請參閱下列主題：

- ABS 函數
- CEILING (或 CEIL) 函數
- FLOOR 函數
- ROUND 函數
- SIGN 函數
- TRUNC 函數

SUPER 資料的考量事項

使用 SUPER 資料時，請考慮以下事項：

- 使用 JDBC 驅動程式 1.2.50 版本、ODBC 驅動程式 1.4.17 或更新版本，以及 Amazon Redshift Python 驅動程式 2.0.872 或更新版本。

如需 JDBC 驅動程式的相關資訊，請參閱設定 JDBC 連線。

如需 ODBC 驅動程式的相關資訊，請參閱設定 ODBC 連線。

- 在 SUPER 範例資料集 中尋找下列主題中使用的結構描述範例。
- 下列主題中使用的所有 SQL 程式碼範例都包含在相同的 S3 字首中以供下載。其中包括資料定義語言 (DDL) 和 COPY 陳述式，以及與 SUPER 搭配使用的某些 TPC-H 修改查詢。

  若要檢視或下載 SQL 檔案，請執行下列其中一項：

  - 下載 SUPER 教學課程 SQL 檔案和 TPC-H 檔案。
  - 使用 Amazon S3 CLI 執行以下命令。您可以使用自己的目標路徑。

```
aws s3 cp s3://redshift-downloads/semistructured/tutorialscripts/semistructured-
tutorial.sql /target/path
aws s3 cp s3://redshift-downloads/semistructured/tutorialscripts/
super_tpch_queries.sql /target/path
```

如需 SUPER 組態的相關資訊，請參閱SUPER 組態。

SUPER 範例資料集

用於擷取和查詢範例的資料表結構描述和資料模型定義如下。

```
/*customer-orders-lineitem*/
CREATE TABLE customer_orders_lineitem
(c_custkey bigint
,c_name varchar
,c_address varchar
,c_nationkey smallint
,c_phone varchar
,c_acctbal decimal(12,2)
,c_mktsegment varchar
,c_comment varchar
,c_orders super
);

/* Datamodel of documents to be stored in c_orders Super column would be as follows*/
ARRAY < STRUCT < o_orderkey:bigint
                          ,o_orderstatus:string
                          ,o_totalprice:double
```

```
                            ,o_orderdate:string
                            ,o_orderpriority:string
                            ,o_clerk:string
                            ,o_shippriority:int
                            ,o_comment:string
                            ,o_lineitems:ARRAY < STRUCT < l_partkey:bigint
                                                          ,l_suppkey:bigint
                                                          ,l_linenumber:int
                                                          ,l_quantity:double
                                                          ,l_extendedprice:double
                                                          ,l_discount:double
                                                          ,l_tax:double
                                                          ,l_returnflag:string
                                                          ,l_linestatus:string
                                                          ,l_shipdate:string
                                                          ,l_commitdate:string
                                                          ,l_receiptdate:string
                                                          ,l_shipinstruct:string
                                                          ,l_shipmode:string
                                                          ,l_comment:string
                                                          > >
                             > >

/*part*/
CREATE TABLE part
(
 p_partkey bigint
 ,p_name varchar
 ,p_mfgr varchar
 ,p_brand varchar
 ,p_type varchar
 ,p_size int
 ,p_container varchar
 ,p_retailprice decimal(12,2)
 ,p_comment varchar
 );

/*region-nations*/
CREATE TABLE region_nations
(
 r_regionkey smallint
 ,r_name varchar
 ,r_comment varchar
 ,r_nations super
```

```
 );

 /* Datamodel of documents to be stored in r_nations Super column would be as follows*/
 ARRAY < STRUCT < n_nationkey:int,n_name:string,n_comment:string > >

/*supplier-partsupp*/
CREATE TABLE supplier_partsupp
(
 s_suppkey bigint
 ,s_name varchar
 ,s_address varchar
 ,s_nationkey smallint
 ,s_phone varchar
 ,s_acctbal double precision
 ,s_comment varchar
 ,s_partsupps super
 );

 /* Datamodel of documents to be stored in s_partsupps Super column would be as
 follows*/
 ARRAY < STRUCT <
 ps_partkey:bigint,ps_availqty:int,ps_supplycost:double,ps_comment:string > >
```

將半結構化資料載入 Amazon Redshift

使用 SUPER 資料類型在 Amazon Redshift 中保留和查詢階層式和一般資料。Amazon Redshift 引入了以 JSON 格式剖析資料並將其轉換為 SUPER 表示的 `json_parse` 函數。Amazon Redshift 還支援使用 COPY 命令載入 SUPER 欄。支援的檔案格式包括 JSON、Avro、文字、逗號分隔值 (CSV) 格式、Parquet 和 ORC。

如需下列範例中所用資料表的詳細資訊，請參閱SUPER 範例資料集。

如需 `json_parse` 函數的詳細資訊，請參閱JSON_PARSE 函數。

SUPER 資料類型的預設編碼是 ZSTD。

將 JSON 文件剖析為 SUPER 欄

您可以使用 `json_parse` 函數將 JSON 資料插入或更新到 SUPER 欄中。該函數剖析 JSON 格式的資料，並將其轉換為 SUPER 資料類型，您可以在 INSERT 或 UPDATE 陳述式中使用。

下列範例將 JSON 資料插入到 SUPER 欄。如果查詢中缺少 json_parse 函數，Amazon Redshift 會將該值視為單一字串，而不是必須剖析的 JSON 格式字串。

如果您更新 SUPER 資料欄，Amazon Redshift 會要求將完整的文件傳遞至欄值。Amazon Redshift 不支援部分更新。

```
INSERT INTO region_nations VALUES(0,
   'lar deposits. blithely final packages cajole. regular waters are final requests.
 regular accounts are according to',
   'AFRICA',
   JSON_PARSE('{"r_nations":[
      {"n_comment":" haggle. carefully final deposits detect slyly agai",
         "n_nationkey":0,
         "n_name":"ALGERIA"
      },
      {"n_comment":"ven packages wake quickly. regu",
         "n_nationkey":5,
         "n_name":"ETHIOPIA"
      },
      {"n_comment":" pending excuses haggle furiously deposits. pending, express pinto
 beans wake fluffily past t",
         "n_nationkey":14,
         "n_name":"KENYA"
      },
      {"n_comment":"rns. blithely bold courts among the closely regular packages use
 furiously bold platelets?",
         "n_nationkey":15,
         "n_name":"MOROCCO"
      },
      {"n_comment":"s. ironic, unusual asymptotes wake blithely r",
         "n_nationkey":16,
         "n_name":"MOZAMBIQUE"
      }
   ]
}'));
```

使用 COPY 在 Amazon Redshift 中載入 SUPER 欄

在下列章節中，您可以了解如何使用 COPY 命令將 JSON 資料載入 Amazon Redshift 的不同方式。

從 JSON 和 Avro 複製資料

透過在 Amazon Redshift 中使用半結構化資料支援，您可以載入 JSON 文件，而不必將其 JSON 結構的屬性分解成多個欄。

Amazon Redshift 提供了兩種方法來使用 COPY 擷取 JSON 文件，即使 JSON 結構完全或部分未知：

1. 使用 noshred 選項將從 JSON 文件衍生的資料儲存到單一 SUPER 資料欄中。當結構描述不知道或預期會改變時，此方法非常有用。因此，這種方法可以更容易地將整個元組儲存在單一 SUPER 欄中。
2. 使用 auto 或 jsonpaths 選項將 JSON 文件分解成多個 Amazon Redshift 欄。屬性可以是 Amazon Redshift 純量或 SUPER 值。

您可以將這些選項與 JSON 或 Avro 格式搭配使用。

分解前的 JSON 物件大小上限為 4 MB。

將 JSON 文件複製到單一 SUPER 資料欄

若要將 JSON 文件複製到單一 SUPER 資料欄中，請使用單一 SUPER 資料欄建立資料表。

```
CREATE TABLE region_nations_noshred (rdata SUPER);
```

將 Amazon S3 的資料複製到單一 SUPER 資料欄。若要將 JSON 來源資料內嵌到單一 SUPER 資料欄中，請在 FORMAT JSON 子句中指定 noshred 選項。

```
COPY region_nations_noshred FROM 's3://redshift-downloads/semistructured/tpch-nested/
data/json/region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 'noshred';
```

COPY 成功擷取 JSON 之後，您的資料表會有一個 rdata SUPER 資料欄，其中包含整個 JSON 物件的資料。擷取的資料會維護 JSON 階層的所有屬性。不過，分葉會轉換成 Amazon Redshift 純量類型，以便有效率地處理查詢。

使用下列查詢擷取原始 JSON 字串。

```
SELECT rdata FROM region_nations_noshred;
```

當 Amazon Redshift 產生 SUPER 資料欄時，它可以透過 JSON 序列化使用 JDBC 做為字串進行存取。如需詳細資訊，請參閱 序列化複雜的巢狀 JSON。

將 JSON 文件複製到多個 SUPER 資料欄

您可以將 JSON 文件分解為多個欄，這些欄可以是 SUPER 資料欄或 Amazon Redshift 純量類型。Amazon Redshift 傳播 JSON 物件的不同部分到不同的欄。

```
CREATE TABLE region_nations
(
 r_regionkey smallint
 ,r_name varchar
 ,r_comment varchar
 ,r_nations super
 );
```

若要將上一個範例的資料複製到資料表中，請在 FORMAT JSON 子句中指定 AUTO 選項，將 JSON 值分割為多個資料欄。COPY 將頂層 JSON 屬性與欄名稱比對，並允許以 SUPER 值的形式擷取巢狀值，例如 JSON 陣列和物件。

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/json/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 'auto';
```

當 JSON 屬性名稱混合使用大寫和小寫時，請在 FORMAT JSON 子句中指定 `auto ignorecase` 選項。如需 COPY 命令的相關資訊，請參閱使用 'auto ignorecase' 選項從 JSON 資料載入。

在某些情況下，欄名稱和 JSON 屬性不相符，或者要載入的屬性巢狀超過層級深度。如果是這樣，請使用 `jsonpaths` 檔案手動將 JSON 屬性對應至 Amazon Redshift 欄。

```
CREATE TABLE nations
(
 regionkey smallint
 ,name varchar
 ,comment super
 ,nations super
 );
```

假設您要將資料載入到欄名不符合 JSON 屬性的表格中。在下列範例中的 `nations` 表就是這種表格。您可以建立 `jsonpaths` 檔案，依據屬性在 `jsonpaths` 陣列中的位置，將屬性路徑對應至表格欄。

```
{"jsonpaths": [
       "$.r_regionkey",
       "$.r_name",
       "$.r_comment",
       "$.r_nations
    ]
}
```

jsonpaths 檔案的位置被用作 FORMAT JSON 的引數。

```
COPY nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/json/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 's3://redshift-downloads/semistructured/tpch-nested/data/jsonpaths/
nations_jsonpaths.json';
```

使用下列查詢來存取顯示資料分佈到多個欄的資料表。SUPER 資料欄使用 JSON 格式進行列印。

```
SELECT r_regionkey,r_name,r_comment,r_nations[0].n_nationkey FROM region_nations ORDER
 BY 1,2,3 LIMIT 1;
```

Jsonpath 黨案將 JSON 文件中的欄位映射到表格欄。您可以擷取其他欄，例如分散索引鍵和排序索引鍵，同時仍將完整的文件載入為 SUPER 欄。以下查詢將完整文件載入到 nations 欄。name 欄是排序索引鍵，而 regionkey 欄是分散索引鍵。

```
CREATE TABLE nations_sorted (
    regionkey smallint,
    name varchar,
    nations super
) DISTKEY(regionkey) SORTKEY(name);
```

根 jsonpath "$" 映射到文件的根目錄，如下所示：

```
{"jsonpaths": [
       "$.r_regionkey",
       "$.r_name",
       "$"
    ]
}
```

jsonpaths 文件的位置被用作 FORMAT JSON 的引數。

```
COPY nations_sorted FROM 's3://redshift-downloads/semistructured/tpch-nested/data/json/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 's3://redshift-downloads/semistructured/tpch-nested/data/jsonpaths/
nations_sorted_jsonpaths.json';
```

從文字和 CSV 複製資料

Amazon Redshift 將文字和 CSV 格式的 SUPER 欄表示為序列化 JSON。SUPER 欄需要有效的 JSON 格式，才能以正確的類型資訊載入。取消物件、陣列、數字、布林值和 null 值的引號。將字串值換成雙引號。SUPER 資料欄會針對文字和 CSV 格式使用標準逸出規則。針對 CSV，會根據 CSV 標準逸出分隔符號。對於文字，如果選擇的分隔符號也可能出現在 SUPER 欄位中，請在 COPY 和 UNLOAD 期間使用 ESCAPE 選項。

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/csv/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT CSV;
```

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/text/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
DELIMITER ','
ESCAPE;
```

從單欄式 Parquet 和 ORC 複製資料

如果您的半結構化或巢狀資料已以 Apache Parquet 或 Apache ORC 格式提供，您可以使用 COPY 命令將資料擷取到 Amazon Redshift 中。

Amazon Redshift 資料表結構應符合 Parquet 或 ORC 檔案的欄數和欄資料類型。藉由在 COPY 命令中指定 SERIALIZETOJSON，您可以將與資料表中的 SUPER 欄對齊的檔案中的任何欄類型載入為 SUPER。這包括結構和陣列類型。

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/
parquet/region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT PARQUET SERIALIZETOJSON;
```

以下範例使用 ORC 格式。

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/orc/
region_nation'
IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT ORC SERIALIZETOJSON;
```

當日期或時間資料類型的屬性在 ORC 中時，Amazon Redshift 在 SUPER 中對它們進行編碼時會將它們轉換為 varchar。

卸載半結構化資料

您可以使用不同格式將含有 SUPER 資料欄的表格卸載到 Amazon S3。

主題

卸載 CSV 或文字格式的半結構化資料

您可以使用逗號分隔值 (CSV) 或文字格式，將包含 SUPER 資料欄的表格卸載到 Amazon S3。使用導覽和解除巢狀化子句的組合，Amazon Redshift 可用 CSV 或文字格式將 SUPER 資料格式的階層式資料卸載到 Amazon S3。隨後，您可以針對未載入的資料建立外部資料表，並使用 Redshift Spectrum 進行查詢。如需使用 UNLOAD 和所需 IAM 許可的詳細資訊，請參閱UNLOAD。

在執行下列範例之前，請先使用將半結構化資料載入 Amazon Redshift中的程序填入 region_nations 表格。如需下列範例中所用資料表的詳細資訊，請參閱SUPER 範例資料集。

下列範例將資料卸載到 Amazon S3。

```
UNLOAD ('SELECT * FROM region_nations')
TO 's3://xxxxxx/'
IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3-Write'
DELIMITER AS '|'
GZIP
ALLOWOVERWRITE;
```

與其他使用者定義字串代表 null 值的資料類型不同，Amazon Redshift 會使用 JSON 格式匯出 SUPER 資料欄，並在 JSON 格式決定時將其表示為 null。因此，SUPER 資料欄會忽略 UNLOAD 命令中使用的 NULL [AS] 選項。

卸載 Parquet 格式的半結構化資料

您可以將具有 SUPER 資料欄的表格以 Parquet 格式卸載到 Amazon S3。Amazon Redshift 將 Parquet 中的 SUPER 欄表示為 JSON 資料類型。這使半結構化資料可以在 Parquet 中表示。您可以使用 Redshift Spectrum 查詢這些欄，或使用 COPY 命令將它們擷取回 Amazon Redshift。如需使用 UNLOAD 和所需 IAM 許可的詳細資訊，請參閱UNLOAD。

下列範例會以 Parquet 格式將資料卸載至 Amazon S3。

```
UNLOAD ('SELECT * FROM region_nations')
TO 's3://xxxxxx/'
IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3-Write'
FORMAT PARQUET;
```

查詢半結構化資料

Amazon Redshift 使用 PartiQL 語言來提供對關聯式、半結構化和巢狀資料的 SQL 相容存取。

PartiQL 使用動態類型進行運作。這種方法可在結構化、半結構化和巢狀資料集的組合上實現直覺式篩選、聯結和彙總。存取巢狀資料時，PartiQL 語法會使用點符號和陣列下標來進行路徑導覽。它也可讓 FROM 子句項目迭代陣列，並用於解除巢狀化操作。接下來，您可以找到不同查詢模式的描述，這些模式結合了使用 SUPER 資料類型與路徑和陣列導覽、解除巢狀化、取消樞紐和聯結。

如需下列範例中所用資料表的詳細資訊，請參閱SUPER 範例資料集。

Navigation (導覽)

Amazon Redshift 使用 PartiQL，分別使用 [...] 括號和點符號啟用導覽到陣列和結構中。此外，您也可以使用點符號將導覽混合到結構中，以及使用括號符號將導覽混合到陣列中。例如，下列範例假設 `c_orders` SUPER 資料欄是具有結構的陣列，且屬性已命名為 `o_orderkey`。

若要擷取 `customer_orders_lineitem` 資料表中的資料，請執行以下命令。以您自己的憑證取代 IAM 角色。

```
COPY customer_orders_lineitem FROM 's3://redshift-downloads/semistructured/tpch-nested/
data/json/customer_orders_lineitem'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 'auto';

SELECT c_orders[0].o_orderkey FROM customer_orders_lineitem;
```

Amazon Redshift 也使用資料表別名做為表示法的字首。以下範例與前面的範例是相同的查詢。

```
SELECT cust.c_orders[0].o_orderkey FROM customer_orders_lineitem AS cust;
```

您可以在所有類型的查詢中使用點和括號符號，例如篩選、聯結和彙總。您可以在查詢中使用這些符號，其中通常有欄參考。下列範例會使用篩選結果的 SELECT 陳述式。

```
SELECT count(*) FROM customer_orders_lineitem WHERE c_orders[0]. o_orderkey IS NOT
 NULL;
```

下列範例會在 GROUP BY 和 ORDER BY 子句中使用括號和點導覽。

```
SELECT c_orders[0].o_orderdate,
       c_orders[0].o_orderstatus,
       count(*)
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderkey IS NOT NULL
GROUP BY c_orders[0].o_orderstatus,
         c_orders[0].o_orderdate
ORDER BY c_orders[0].o_orderdate;
```

解除巢狀化查詢

為了解除巢狀化查詢，Amazon Redshift 使用 PartiQL 語法來迭代 SUPER 陣列。它透過使用查詢的 FROM 子句導覽陣列來實現此目的。使用上一個範例，下列範例會迭代 c_orders 的屬性值。

```
SELECT c.*, o FROM customer_orders_lineitem c, c.c_orders o;
```

解除巢狀化語法是 FROM 子句的擴充功能。在標準 SQL 中，FROM 子句 x （AS） y 代表 y 迭代關係 x 的每個元組。在這種情況下，x 指的是關係，而 y 指的是關係 x 的別名。同樣地，使用 FROM 子句項目 x （AS） y 解除巢狀化的 PartiQL 語法表示 y 會迭代 (SUPER) 陣列運算式 x 中的每個 (SUPER) 值。在這種情況下，x 是 SUPER 運算式，而 y 是 x 的別名。

左運算元也可以使用點和括號符號進行常規導覽。在上一個範例中，customer_orders_lineitem c 是 customer_order_lineitem 基底資料表上的迭代，而 c.c_orders o 是 c.c_orders 陣列上的迭代。若要迭代 o_lineitems 屬性 (即陣列中的陣列)，您可以新增多個子句。

```
SELECT c.*, o, l FROM customer_orders_lineitem c, c.c_orders o, o.o_lineitems l;
```

使用 AT 關鍵字迭代陣列時，Amazon Redshift 也支援陣列索引。子句 x AS y AT z 會迭代陣列 x 並產生做為陣列索引的欄位 z,。下列範例顯示陣列索引的運作方式。

```
SELECT c_name,
       orders.o_orderkey AS orderkey,
       index AS orderkey_index
FROM customer_orders_lineitem c, c.c_orders AS orders AT index
ORDER BY orderkey_index;

c_name             | orderkey | orderkey_index
-------------------+----------+----------------
Customer#000008251 | 3020007  |        0
Customer#000009452 | 4043971  |        0
  (2 rows)
```

下列範例會迭代純量陣列。

```
CREATE TABLE bar AS SELECT json_parse('{"scalar_array": [1, 2.3, 45000000]}') AS data;

SELECT index, element FROM bar AS b, b.data.scalar_array AS element AT index;

 index | element
-------+----------
     0 | 1
     1 | 2.3
     2 | 45000000
(3 rows)
```

下列範例會迭代多個層級的陣列。這個範例會使用多個解除巢狀化子句來迭代到最內層的陣列。f.multi_level_array AS 陣列會迭代 multi_level_array。陣列 AS 元素是在 multi_level_array 內的陣列的迭代。

```
CREATE TABLE foo AS SELECT json_parse('[[1.1, 1.2], [2.1, 2.2], [3.1, 3.2]]') AS
 multi_level_array;

SELECT array, element FROM foo AS f, f.multi_level_array AS array, array AS element;

   array   | element
-----------+---------
 [1.1,1.2] | 1.1
 [1.1,1.2] | 1.2
 [2.1,2.2] | 2.1
```

```
 [2.1,2.2] | 2.2
 [3.1,3.2] | 3.1
 [3.1,3.2] | 3.2
(6 rows)
```

如需 FROM 子句的相關資訊，請參閱FROM 子句。

物件取消樞紐

若要執行物件取消樞紐，Amazon Redshift 會使用 PartiQL 語法來迭代 SUPER 物件。它使用具有 UNPIVOT 關鍵字的查詢的 FROM 子句來執行此操作。在這種情況下，表達式是對`c.c_orders[0]`象。範例查詢會重複執行物件傳回的每個屬性。

```
SELECT attr as attribute_name, json_typeof(val) as value_type
FROM customer_orders_lineitem c, UNPIVOT c.c_orders[0] AS val AT attr
WHERE c_custkey = 9451;

 attribute_name  | value_type
-----------------+------------
 o_orderstatus   | string
 o_clerk         | string
 o_lineitems     | array
 o_orderdate     | string
 o_shippriority  | number
 o_totalprice    | number
 o_orderkey      | number
 o_comment       | string
 o_orderpriority | string
(9 rows)
```

如同解除巢狀化結構，取消樞紐語法也是 FROM 子句的擴充功能。不同之處在於，取消樞紐的語法使用 UNPIVOT 關鍵字來指示它正在迭代物件而不是陣列。它使用 AS `value_alias` 迭代物件內的所有值，並使用 AT `attribute_alias` 迭代所有屬性。考慮下面的語法片段：

```
UNPIVOT expression AS value_alias [ AT attribute_alias ]
```

Amazon Redshift 支援在單一 FROM 子句中使用物件解除樞紐和陣列解除巢狀，如下所示：

```
SELECT attr as attribute_name, val as object_value
FROM customer_orders_lineitem c, c.c_orders AS o, UNPIVOT o AS val AT attr
WHERE c_custkey = 9451;
```

當您使用物件取消樞紐時，Amazon Redshift 不支援相關的取消樞紐。具體來說，假設您有多個在不同查詢層級中取消樞紐的範例，而內部未樞紐分析會參考外部層級的範例。Amazon Redshift 不支援這種類型的多重取消樞紐。

如需 FROM 子句的相關資訊，請參閱FROM 子句。如需示範如何使用 PIVOT 和 UNPIVOT 查詢結構化資料的範例，請參閱PIVOT 和 UNPIVOT 範例。

動態類型

動態類型不需要明確轉換從點和括號路徑中擷取的資料。Amazon Redshift 會使用動態類型來處理無結構描述的 SUPER 資料，無需在查詢中使用資料類型之前先宣告資料類型。動態類型會使用導覽至 SUPER 資料欄的結果，而不必明確地將它們轉換為 Amazon Redshift 類型。動態類型在聯結和 GROUP BY 子句中最有用。下列範例使用 SELECT 陳述式，該陳述式不需要將點和方括號運算式明確轉換為一般的 Amazon Redshift 類型。如需類型相容性和轉換的相關資訊，請參閱類型相容性與轉換。

```
SELECT c_orders[0].o_orderkey
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderstatus = 'P';
```

當 c_orders[0].o_orderstatus 是字串 'P' 時，此查詢中的等號計算結果為 `true`。在所有其他情況下，等號評估為 `false`，包括相等參數是不同類型的情況。

動態和靜態類型

如果不使用動態類型，則無法確定 c_orders[0].o_orderstatus 是字串、整數還是結構。您只能確定 c_orders[0].o_orderstatus 是 SUPER 資料類型，它可以是 Amazon Redshift 純量、陣列或結構。c_orders[0].o_orderstatus 的靜態類型是 SUPER 資料類型。通常，類型隱含地是 SQL 中的靜態類型。

Amazon Redshift 使用動態類型來處理無結構描述資料。當查詢評估資料時，c_orders[0].o_orderstatus 結果是特定類型。例如，對 customer_orders_lineitem 的第一則記錄評估 c_orders[0].o_orderstatus 可能會得到整數。對第二則記錄進行評估可能會得到字串。這些是運算式的動態類型。

將 SQL 運算子或函數搭配具有動態類型的點和括號運算式使用時，Amazon Redshift 會產生類似於將標準 SQL 運算子或函數搭配個別靜態類型使用的結果。在此範例中，當路徑運算式的動態類型是字串時，與字串 'P' 的比較是有意義的。只要 c_orders[0].o_orderstatus 的動態類型是字串以外的任何其他資料類型，則等式傳回 false。當使用類型錯誤的引數時，其他函數傳回 null。

下列範例使用靜態類型撰寫先前的查詢：

```
SELECT c_custkey
FROM customer_orders_lineitem
WHERE CASE WHEN JSON_TYPEOF(c_orders[0].o_orderstatus) = 'string'
           THEN c_orders[0].o_orderstatus::VARCHAR = 'P'
           ELSE FALSE END;
```

請注意相等述詞和比較述詞之間的以下區別。在前面的範例中，如果您以述詞取代相等 less-than-or-equal述詞，語意會產生 null 而不是 false。

```
SELECT c_orders[0]. o_orderkey
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderstatus <= 'P';
```

在此範例中，如果 c_orders[0].o_orderstatus 是字串，且按字母順序等於或小於 'P'，則 Amazon Redshift 傳回 true。如果它的字母順序大於 'P'，則 Amazon Redshift 傳回 false。但是，如果 c_orders[0].o_orderstatus 不是字串，Amazon Redshift 將傳回 null，因為 Amazon Redshift 無法比較不同類型的值，如以下查詢所示：

```
SELECT c_custkey
FROM customer_orders_lineitem
WHERE CASE WHEN JSON_TYPEOF(c_orders[0].o_orderstatus) = 'string'
           THEN c_orders[0].o_orderstatus::VARCHAR <= 'P'
           ELSE NULL END;
```

動態類型並不排除具有最低可比性的類型比較。例如，您可以將 CHAR 和 VARCHAR Amazon Redshift 純量類型轉換為 SUPER。它們與字串相當，包括忽略類似 Amazon Redshift CHAR 和 VARCHAR 類型的結尾空白字元。同樣地，整數、小數和浮點值與 SUPER 值具有可比性。特別是對於小數欄，每個值也可以有不同的小數位數。Amazon Redshift 仍將它們視為動態類型。

Amazon Redshift 也支援評估為深度相等的物件和陣列的相等性，例如深入評估物件或陣列並比較所有屬性。請謹慎使用深度相等，因為執行深度相等的過程可能非常耗時。

使用動態類型進行聯結

對於聯結，動態類型會自動比對具有不同動態類型的值，而無需執行長時間的 CASE WHEN 分析以找出可能出現的資料類型。例如，假設您的組織隨著時間變更了其用於部分索引鍵的格式。

一開始發行的整數部分索引鍵更換為字串部分索引鍵，例如 'A55'，後來再次更換為陣列部分索引鍵，例如組合字串和數字的 ['X'，10]。Amazon Redshift 不需要對部分索引鍵執行冗長的案例分析，而且可以使用聯結，如下列範例所示。

```
SELECT c.c_name
    ,l.l_extendedprice
    ,l.l_discount
FROM customer_orders_lineitem c
    ,c.c_orders o
    ,o.o_lineitems l
    ,supplier_partsupp s
    ,s.s_partsupps ps
WHERE l.l_partkey = ps.ps_partkey
AND c.c_nationkey = s.s_nationkey
ORDER BY c.c_name;
```

下列範例顯示如果不使用動態類型，相同查詢可能會有多複雜且效率低落：

```
SELECT c.c_name
    ,l.l_extendedprice
    ,l.l_discount
FROM customer_orders_lineitem c
    ,c.c_orders o
    ,o.o_lineitems l
    ,supplier_partsupp s
    ,s.s_partsupps ps
WHERE CASE WHEN IS_INTEGER(l.l_partkey) AND IS_INTEGER(ps.ps_partkey)
           THEN l.l_partkey::integer = ps.ps_partkey::integer
           WHEN IS_VARCHAR(l.l_partkey) AND IS_VARCHAR(ps.ps_partkey)
           THEN l.l_partkey::varchar = ps.ps_partkey::varchar
           WHEN IS_ARRAY(l.l_partkey) AND IS_ARRAY(ps.ps_partkey)
                AND IS_VARCHAR(l.l_partkey[0]) AND IS_VARCHAR(ps.ps_partkey[0])
                AND IS_INTEGER(l.l_partkey[1]) AND IS_INTEGER(ps.ps_partkey[1])
           THEN l.l_partkey[0]::varchar = ps.ps_partkey[0]::varchar
                AND l.l_partkey[1]::integer = ps.ps_partkey[1]::integer
           ELSE FALSE END
AND c.c_nationkey = s.s_nationkey
ORDER BY c.c_name;
```

寬鬆的語義

預設情況下，導覽無效時 SUPER 值的導覽操作會傳回 null，而不是傳回錯誤。如果 SUPER 值不是物件，或者 SUPER 值是物件但不包含查詢中使用的屬性名稱，則物件導覽無效。例如，下列查詢會存取 SUPER 資料欄 cdata 中無效的屬性名稱：

```
SELECT c.c_orders.something FROM customer_orders_lineitem c;
```

如果 SUPER 值不是陣列或陣列索引超出邊界，則陣列導覽傳回 null。下列查詢會傳回 null，因為 c_orders[1][1] 超出範圍。

```
SELECT c.c_orders[1][1] FROM customer_orders_lineitem c;
```

寬鬆的語義在使用動態類型轉換 SUPER 值時特別有用。如果轉換無效，將 SUPER 值轉換為錯誤類型會傳回 null，而不是錯誤。例如，以下查詢傳回 null，因為它無法將物件屬性 o_orderstatus 的字串值 'Good' 轉換為 INTEGER。Amazon Redshift 對於 VARCHAR 到 INTEGER 轉換傳回錯誤，但對於 SUPER 轉換則不會傳回錯誤。

```
SELECT c.c_orders.o_orderstatus::integer FROM customer_orders_lineitem c;
```

自我檢查的種類

SUPER 資料欄支援傳回動態類型和 SUPER 值的其他類型資訊的檢查函數。最常見的範例是 JSON_TYPEOF 純量函數，此函數會根據 SUPER 值的動態類型，傳回含有布林值、數字、字串、物件、陣列或 null 的 VARCHAR。Amazon Redshift 支援 SUPER 資料欄的下列布林函數：

- DECIMAL_PRECISION
- DECIMAL_SCALE
- IS_ARRAY
- IS_BIGINT
- IS_CHAR
- IS_DECIMAL
- IS_FLOAT
- IS_INTEGER
- IS_OBJECT
- IS_SCALAR

- IS_SMALLINT
- IS_VARCHAR
- JSON_TYPEOF

如果輸入值為 null，所有這些函數傳回 false。IS_SCALAR、IS_OBJECT 和 IS_ARRAY 是互斥的，涵蓋除 null 之外的所有可能值。

為了推論資料的對應類型，Amazon Redshift 使用 JSON_TYPEOF 函數傳回 SUPER 值 (頂層) 的類型，如下列範例所示：

```
SELECT JSON_TYPEOF(r_nations) FROM region_nations;
 json_typeof
 -------------
 array
(1 row)
```

```
SELECT JSON_TYPEOF(r_nations[0].n_nationkey) FROM region_nations;
 json_typeof
 -------------
 number
```

Amazon Redshift 將其視為單一長字串，類似於將此值插入 VARCHAR 欄而不是 SUPER。由於該欄是 SUPER，因此單一字串仍然是有效的 SUPER 值，並且在 JSON_TYPEOF 中註明了差異：

```
SELECT IS_VARCHAR(r_nations[0].n_name) FROM region_nations;
 is_varchar
-------------
 true
(1 row)
```

```
SELECT r_nations[4].n_name FROM region_nations
WHERE CASE WHEN IS_INTEGER(r_nations[4].n_nationkey)
           THEN r_nations[4].n_nationkey::INTEGER = 15
           ELSE false END;
```

排序依據

Amazon Redshift 不會定義具有不同動態類型的值之間的 SUPER 比較。字串形式的 SUPER 值既不小於也不大於數字形式的 SUPER 值。為了將 ORDER BY 子句與 SUPER 欄結合使用，Amazon

Redshift 定義了當 Amazon Redshift 使用 ORDER BY 子句對 SUPER 值進行排名時要觀察的不同類型之間的總排序。動態類型之間的順序是布林、數字、字串、陣列、物件。以下範例顯示了不同類型的順序：

```
INSERT INTO region_nations VALUES
(100,'name1','comment1','AWS'),
(200,'name2','comment2',1),
(300,'name3','comment3',ARRAY(1, 'abc', null)),
(400,'name4','comment4',-2.5),
(500,'name5','comment5','Amazon');

SELECT r_nations FROM region_nations order by r_nations;

r_nations
----------------
 -2.5
 1
 "Amazon"
 "AWS"
 [1,"abc",null]
(5 rows)
```

如需 ORDER BY 子句的相關資訊，請參閱ORDER BY 子句。

運算子和函數

Amazon Redshift 提供 SUPER 運算子和函數的以下功能支援。

算術運算子

SUPER 值支援使用動態類型的所有基本算術運算子 +、-、*、/、%。運算的結果類型會保持為 SUPER。對於所有運算子，除了二元運算子 + 之外，輸入運算元必須是數字。否則，Amazon Redshift 傳回 null。當 Amazon Redshift 執行這些運算子且動態類型不會變更時，將保留小數和浮點值之間的差異。但是，當您使用乘法和除法時，小數位數會發生變化。算術溢位仍然會導致查詢錯誤，它們不會變更為 null。如果輸入是數字，則二元運算子 + 執行加法；如果輸入是字串，則執行串連。如果一個運算元是一個字串，而另一個運算元是一個數字，則結果為 null。如果 SUPER 值不是數字，一元前綴運算子 + 和 - 將傳回 null，如以下範例所示：

```
SELECT (c_orders[0]. o_orderkey + 0.5) * c_orders[0]. o_orderkey / 10 AS math FROM
 customer_orders_lineitem;
           math
```

```
---------------------------
 1757958232200.1500
(1 row)
```

動態類型允許 SUPER 中的小數值具有不同的小數位數。Amazon Redshift 會將十進位值視為不同的靜態類型，並允許所有數學運算。Amazon Redshift 會根據運算元的比例，動態計算結果小數位數。如果其中一個運算元是浮點數，則 Amazon Redshift 會將另一個運算元提升為浮點數並產生浮點數結果。

算術函數

Amazon Redshift 支援 SUPER 欄的以下算術函數。如果輸入不是數字，以下項目會傳回 null：

- FLOOR。如需詳細資訊，請參閱 FLOOR 函數。
- CEIL 和 CEILING。如需詳細資訊，請參閱 CEILING (或 CEIL) 函數。
- ROUND。如需詳細資訊，請參閱 ROUND 函數。
- TRUNC。如需詳細資訊，請參閱 TRUNC 函數。
- ABS。如需詳細資訊，請參閱 ABS 函數。

下列範例會使用算術函數來查詢資料：

```
SELECT x, FLOOR(x), CEIL(x), ROUND(x)
FROM (
    SELECT (c_orders[0]. o_orderkey + 0.5) * c_orders[0].o_orderkey / 10 AS x
    FROM customer_orders_lineitem
    );

         x          |     floor     |     ceil      |     round
--------------------+---------------+---------------+---------------
 1389636795898.0500  | 1389636795898  | 1389636795899  | 1389636795898
```

ABS 函數會保留輸入小數的小數位數，而 FLOOR、CEIL。ROUND 會消除輸入小數的小數位數。

陣列函數

Amazon Redshift 支援以下陣列組合和實用函數
array、array_concat、subarray、array_flatten、get_array_length 和 split_to_array。

您可以使用 ARRAY 函數 (包括其他 SUPER 值)，從 Amazon Redshift 資料類型中的值建構 SUPER 陣列。以下範例使用可變參數函數 ARRAY：

```
SELECT ARRAY(1, c.c_custkey, NULL, c.c_name, 'abc') FROM customer_orders_lineitem c;
                                array
 ------------------------------------------------------
[1,8401,null,""Customer#000008401"",""abc""]
[1,9452,null,""Customer#000009452"",""abc""]
[1,9451,null,""Customer#000009451"",""abc""]
[1,8251,null,""Customer#000008251"",""abc""]
[1,5851,null,""Customer#000005851"",""abc""]
(5 rows)
```

下列範例使用 ARRAY_CONCAT 函數進行陣列串連：

```
SELECT ARRAY_CONCAT(JSON_PARSE('[10001,10002]'),JSON_PARSE('[10003,10004]'));

             array_concat
-------------------------------------
 [10001,10002,10003,10004]
(1 row)
```

下列範例使用 SUBARRAY 函數進行陣列操作，該函數會傳回輸入陣列的子集。

```
SELECT SUBARRAY(ARRAY('a', 'b', 'c', 'd', 'e', 'f'), 2, 3);

   subarray
---------------
 ["c","d","e"]
(1 row))
```

下列範例使用 ARRAY_FLATTEN 將多個層級的陣列合併為一個陣列：

```
SELECT x, ARRAY_FLATTEN(x) FROM (SELECT ARRAY(1, ARRAY(2, ARRAY(3, ARRAY())))) AS x);

     x           | array_flatten
 ----------------+---------------
 [1,[2,[3,[]]]]  | [1,2,3]
(1 row)
```

陣列函數 ARRAY_CONCAT 和 ARRAY_FLATTEN 使用動態類型規則。如果輸入不是陣列，它們會傳回 null 而不是錯誤。GET_ARRAY_LENGTH 函數傳回給定的物件或陣列路徑的 SUPER 陣列的長度。

```
SELECT c_name
```

```
FROM customer_orders_lineitem
WHERE GET_ARRAY_LENGTH(c_orders) = (
    SELECT MAX(GET_ARRAY_LENGTH(c_orders))
    FROM customer_orders_lineitem
    );
```

下列範例使用 SPLIT_TO_ARRAY 將字串分割為字串陣列。該函數使用分隔符號做為選用參數。如果沒有分隔符號，則預設為逗號。

```
SELECT SPLIT_TO_ARRAY('12|345|6789', '|');

   split_to_array
---------------------
 ["12","345","6789"]
(1 row)
```

SUPER 組態

當您使用 Amazon Redshift SUPER 資料類型和 PartiQL 時，請注意 SUPER 組態的下列考量事項。

SUPER 的寬鬆和嚴格模式

當您查詢 SUPER 資料時，路徑運算式可能不符合實際的 SUPER 資料結構。如果您嘗試存取物件或陣列元素的不存在成員，且您的查詢是在預設的寬鬆模式下執行，Amazon Redshift 會傳回 NULL 值。如果以嚴格模式執行查詢，則 Amazon Redshift 會傳回錯誤訊息。下列工作階段參數可以設定為開啟或關閉寬鬆模式。

下列範例會使用工作階段參數來啟用寬鬆模式。

```
SET navigate_super_null_on_error=ON;  --default lax mode for navigation

SET cast_super_null_on_error=ON;  --default lax mode for casting

SET parse_super_null_on_error=OFF;  --default strict mode for ingestion
```

存取具有大寫和混合大小寫欄位名稱或屬性的 JSON 欄位

當您的 JSON 屬性名稱為大寫或混合大小寫時，您必須能夠以區分大小寫的方式瀏覽 SUPER 類型結構。要做到這一點，你可以將 enable_case_sensitive_identifier

設定為 TRUE，並用雙引號將大寫和混合大小寫的屬性名稱括起來。您也可以將 enable_case_sensitive_super_attribute 設定為 TRUE。在這種情況下，您可以在查詢中使用大寫和混合大小寫的屬性名稱，而無需將它們用雙引號引起來。

下列範例說明如何設定 enable_case_sensitive_identifier 來查詢資料。

```
SET enable_case_sensitive_identifier to TRUE;

-- Accessing JSON attribute names with uppercase and mixedcase names
SELECT json_table.data."ITEMS"."Name",
       json_table.data."price"
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;

 Name | price
------+-------
 "TV" | 345
(1 row)

RESET enable_case_sensitive_identifier;

-- After resetting the above configuration, the following query accessing JSON
 attribute names with uppercase and mixedcase names should return null (if in lax
 mode).
SELECT json_table.data."ITEMS"."Name",
       json_table.data."price"
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;

 name | price
------+-------
      | 345
(1 row)
```

下列範例說明如何設定 enable_case_sensitive_super_attribute 來查詢資料。

```
SET enable_case_sensitive_super_attribute to TRUE;
 -- Accessing JSON attribute names with uppercase and mixedcase names

SELECT json_table.data.ITEMS.Name,
       json_table.data.price
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;
```

```
 name | price
 -----+-------
 "TV" | 345
(1 row)

 RESET enable_case_sensitive_super_attribute;

 -- After resetting enable_case_sensitive_super_attribute, the query now returns NULL
 for ITEMS.Name (if in lax mode).

SELECT json_table.data.ITEMS.Name,
       json_table.data.price
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;

 name | price
 -----+-------
      | 345
(1 row)
```

SUPER 的剖析選項

當您使用 JSON_PARSE 函數將 JSON 字串剖析為 SUPER 值時，適用某些限制：

- 相同的屬性名稱不能出現在相同物件中，但可以出現在巢狀物件中。`json_parse_dedup_attributes` 組態選項允許 JSON_PARSE 僅保留最後一次出現的重複屬性，而不是傳回錯誤。
- 字串值不得超過 65535 位元組的系統最大 varchar 大小。`json_parse_truncate_strings` 組態選項允許 JSON_PARSE() 自動截斷長度超過此限制的字串而不傳回錯誤。此行為只會影響字串值，而不會影響屬性名稱。

如需 JSON_PARSE 函數的相關資訊，請參閱JSON_PARSE 函數。

下列範例顯示如何將 `json_parse_dedup_attributes` 組態選項設定為傳回重複屬性錯誤的預設行為。

```
SET json_parse_dedup_attributes=OFF;  --default behavior of returning error instead of
 de-duplicating attributes
```

下列範例顯示如何設定 json_parse_truncate_strings 組態選項，針對長度超過此限制的字串傳回錯誤的預設行為。

```
SET json_parse_truncate_strings=OFF;  --default behavior of returning error instead of
 truncating strings
```

限制

使用 SUPER 資料類型時，請考慮下列限制：

- 您無法將 SUPER 欄定義為分散或排序索引鍵。
- 一個單獨的 SUPER 對象最多可以容納 16 MB 的數據。
- SUPER 物件中的個別值會限制為對應 Amazon Redshift 類型的最大長度。例如，載入至 SUPER 的單一字串值會限制為 65535 個位元組的最大 VARCHAR 長度。
- 您無法在 SUPER 欄上執行部分更新或轉換操作。
- 您無法在右聯結或完整外部聯結中使用 SUPER 資料類型及其別名。
- SUPER 資料類型不支援 XML 做為傳入或傳出序列化格式。
- 在參考資料表變數進行解除巢狀化的子查詢 (無論是否相關) 的 FROM 子句中，查詢只能參考其父資料表，而不能參考其他資料表。
- 轉換限制

  SUPER 值可與其他資料類型互相轉換，但下列情況除外：

  - Amazon Redshift 不區分 0 等級的整數和小數。
  - 如果小數位數不為零，SUPER 資料類型與其他 Amazon Redshift 資料類型具有相同的行為，不同之處在於 Amazon Redshift 會將 SUPER 相關錯誤轉換為 null，如下列範例所示。

    ```
    SELECT 5::bool;
     bool
    -------
     True
    (1 row)

    SELECT 5::decimal::bool;
    ERROR:  cannot cast type numeric to boolean

    SELECT 5::super::bool;
     bool
    ```

```
-------
 True
(1 row)

SELECT 5.0::bool;
ERROR:  cannot cast type numeric to boolean

SELECT 5.0::super::bool;
 bool
-------
(1 row)
```

- Amazon Redshift 不會將日期和時間類型轉換為 SUPER 資料類型。Amazon Redshift 只能從 SUPER 資料類型轉換日期和時間資料類型，如下範例所示。

```
SELECT o.o_orderdate FROM customer_orders_lineitem c,c.c_orders o;
  order_date
------------------
 "2001-09-08"
(1 row)


SELECT JSON_TYPEOF(o.o_orderdate) FROM customer_orders_lineitem c,c.c_orders o;
 json_typeof
-----------------
 string
(1 row)


SELECT o.o_orderdate::date FROM customer_orders_lineitem c,c.c_orders o;
 order_date
----------------
 2001-09-08
(1 row)


--date/time cannot be cast to super
SELECT '2019-09-09'::date::super;
ERROR:  cannot cast type date to super
```

- 從非純量值 (物件和陣列) 轉換為字串會傳回 NULL。若想正確序列化這些非純量值，請不要轉換它們。而是使用 json_serialize 轉換非純量值。json_serialize 函數傳回 varchar。一般

而言，您不需要將非純量值轉換為 varchar，因為 Amazon Redshift 會隱含序列化，如下列第一個範例所示。

```
SELECT r_nations FROM region_nations WHERE r_regionkey=300;
   r_nations
----------------
 [1,"abc",null]
(1 row)

SELECT r_nations::varchar FROM region_nations WHERE r_regionkey=300;
 r_nations
-------------
(1 row)

SELECT JSON_SERIALIZE(r_nations) FROM region_nations WHERE r_regionkey=300;
 json_serialize
-----------------
 [1,"abc",null]
(1 row)
```

  - 對於不區分大小寫的資料庫，Amazon Redshift 不支援 SUPER 資料類型。對於不區分大小寫的欄，Amazon Redshift 不會將它們轉換為 SUPER 類型。因此，Amazon Redshift 不支援 SUPER 欄與觸發轉換的不區分大小寫欄互動。
- Amazon Redshift 不支援子查詢中的揮發性函數，例如 RANDOM ( ) 或 TIMEOFDAY ( )，這些子查詢將外部資料表或 IN 函數的左側 (LHS) 與此類子查詢解除巢狀化。

使用 SUPER 資料類型搭配具體化視觀表

Amazon Redshift 擴展了具體化視觀表的功能，以便在具體化視觀表中處理 SUPER 資料類型和 PartiQL。SQL 和 PartiQL 查詢可以使用累加式具體化視觀表來預先計算。如需具體化視觀表的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

將無結構描述和半結構化資料儲存到 SUPER 之後，您可以使用 PartiQL 具體化視觀表來自我檢查資料，並將它們分解成具體化視觀表。

加速 PartiQL 查詢

您可以使用具體化視觀表來加速導覽和/或解除巢狀化 SUPER 欄中階層式資料的 PartiQL 查詢。建立一或多個具體化視觀表，將 SUPER 值分解成多個欄，並利用 Amazon Redshift 分析查詢的單欄式組織。因此，查詢會使用具體化視觀表。

具體化視觀表本質上是提取和標準化巢狀資料。標準化程度取決於您將 SUPER 資料轉換為傳統單欄式資料所付出的努力。

以具體化視觀表分解為 SUPER 欄

下列範例顯示的具體化視觀表會將巢狀資料分解，產生的欄仍為 SUPER 資料類型。

```
SELECT c.c_name, o.o_orderstatus
FROM customer_orders_lineitem c, c.c_orders o;
```

下列範例顯示一個具體化視觀表，該視觀表會根據分解的資料建立傳統的 Amazon Redshift 純量欄。

```
SELECT c.c_name, c.c_orders[0].o_totalprice
FROM customer_orders_lineitem c;
```

您可以建立單一具體化視觀表 super_mv 來加速這兩項查詢。

要回答第一個查詢，您必須具體化屬性 o_orderstatus。您可以省略 c_name 屬性，因為它不涉及巢狀導覽或解除巢狀化。您也必須在具體化視觀表中包含 customer_orders_lineitem 的屬性 c_custkey，以便能夠將基底資料表聯結至具體化視觀表。

要回答第二個查詢，您還必須具體化屬性 o_totalprice 和 c_orders 的陣列索引 o_idx。因此，您可以存取 c_orders 的索引 0。

```
CREATE MATERIALIZED VIEW super_mv distkey(c_custkey) sortkey(c_custkey) AS (
  SELECT c_custkey, o.o_orderstatus, o.o_totalprice, o_idx
  FROM customer_orders_lineitem c, c.c_orders o AT o_idx
);
```

具體化視觀表 super_mv 的屬性 o_orderstatus 和 o_totalprice 都是 SUPER。

具體化視觀表 super_mv 會在基底資料表 customer_orders_lineitem 發生變更時，累加式重新整理。

```
REFRESH MATERIALIZED VIEW super_mv;
INFO: Materialized view super_mv was incrementally updated successfully.
```

若要將第一個 PartiQL 查詢重寫為常規 SQL 查詢，請將 customer_orders_lineitem 與 super_mv 聯結起來，如下所示。

```
SELECT c.c_name, v.o_orderstatus
```

```
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey;
```

同樣地，您可以重新撰寫第二個 PartiQL 查詢。下列範例會在 o_idx = 0 上使用篩選器。

```
SELECT c.c_name, v.o_totalprice
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey
WHERE v.o_idx = 0;
```

在 CREATE MATERIALIZED VIEW 命令中，指定 c_custkey 做為 super_mv 的分散索引鍵和排序索引鍵。Amazon Redshift 執行高效率的合併聯結，假設 c_custkey 也是 customer_orders_lineitem 的分散索引鍵和排序索引鍵。如果不是這種情況，您可以指定 c_custkey 做為 customer_orders_lineitem 的排序索引鍵和分散索引鍵，如下所示。

```
ALTER TABLE customer_orders_lineitem
ALTER DISTKEY c_custkey, ALTER SORTKEY (c_custkey);
```

使用 EXPLAIN 陳述式來確認 Amazon Redshift 是否對重寫的查詢執行合併聯結。

```
EXPLAIN
      SELECT c.c_name, v.o_orderstatus
      FROM customer_orders_lineitem c JOIN super_mv v ON c.c_custkey = v.c_custkey;

      QUERY PLAN

 ------------------------------------------------------------------------------------------------
      XN Merge Join DS_DIST_NONE  (cost=0.00..34701.82 rows=1470776 width=27)
      Merge Cond: ("outer".c_custkey = "inner".c_custkey)
      ->  XN Seq Scan on mv_tbl__super_mv__0 derived_table2  (cost=0.00..14999.86
 rows=1499986 width=13)
      ->  XN Seq Scan on customer_orders_lineitem c  (cost=0.00..999.96 rows=99996
 width=30)
      (4 rows)
```

從分解的資料建立 Amazon Redshift 純量欄

儲存在 SUPER 中的無結構描述資料可能會影響 Amazon Redshift 的效能。例如，篩選述詞或聯結條件做為範圍限制掃描可能無法有效地使用區域地圖。使用者和 BI 工具可以使用具體化視觀表做為資料的常規表示形式，並提高分析查詢的效能。

下列查詢會掃瞄具體化視觀表 super_mv 和篩選 o_orderstatus。

```
SELECT c.c_name, v.o_totalprice
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey
WHERE v.o_orderstatus = 'F';
```

檢查 stl_scan 以確認 Amazon Redshift 無法在 o_orderstatus 範圍限制掃描上有效地使用區域地圖。

```
SELECT slice, is_rrscan FROM stl_scan
WHERE query = pg_last_query_id() AND perm_table_name LIKE '%super_mv%';

 slice | is_rrscan
-------+-----------
     0 | f
     1 | f
     5 | f
     4 | f
     2 | f
     3 | f
(6 rows)
```

下列範例會調整具體化視觀表 super_mv，從分解的資料建立純量欄。在這種情況下，Amazon Redshift 會將 o_orderstatus 從 SUPER 轉換為 VARCHAR。此外，請指定 o_orderstatus 做為 super_mv 的排序索引鍵。

```
CREATE MATERIALIZED VIEW super_mv distkey(c_custkey) sortkey(c_custkey, o_orderstatus)
 AS (
  SELECT c_custkey, o.o_orderstatus::VARCHAR AS o_orderstatus, o.o_totalprice, o_idx
  FROM customer_orders_lineitem c, c.c_orders o AT o_idx
);
```

重新執行查詢之後，請確認 Amazon Redshift 現在可以使用區域地圖。

```
SELECT v.o_totalprice
FROM super_mv v
WHERE v.o_orderstatus = 'F';
```

您可以驗證範圍限制掃描現在使用區域地圖，如下所示。

```
SELECT slice, is_rrscan FROM stl_scan
WHERE query = pg_last_query_id() AND perm_table_name LIKE '%super_mv%';

 slice | is_rrscan
-------+-----------
     0 | t
     1 | t
     2 | t
     3 | t
     4 | t
     5 | t
(6 rows)
```

將 SUPER 資料類型與具體化視觀表搭配使用的限制

將 SUPER 資料類型與具體化視觀表搭配使用時，請注意下列限制。

Amazon Redshift 中的具體化視觀表對於 PartiQL 或 SUPER 沒有任何特定的限制。

如需有關建立具體化視觀表時有何限制的資訊，請參閱限制。

如需有關累加式重新整理具體化視觀表之一般 SQL 限制的資訊，請參閱增量重新整理的限制。

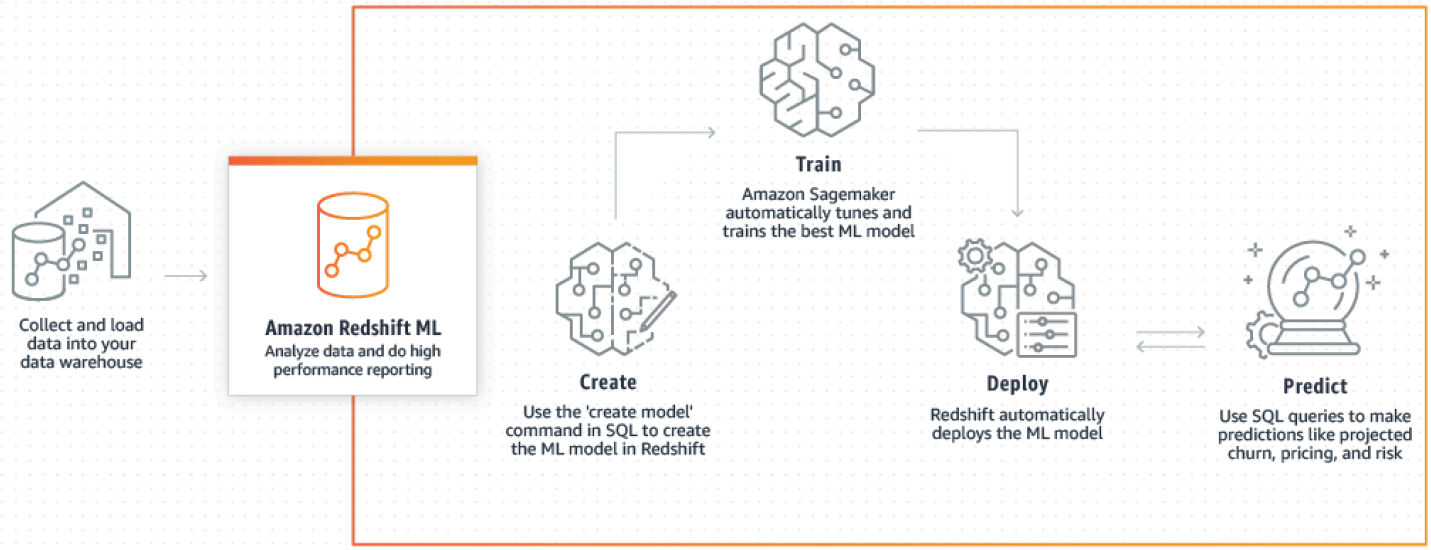
在 Amazon Redshift 中使用機器學習

Amazon Redshift 機器學習 (Amazon Redshift ML) 是一種強大的雲端型服務，可讓各種技能等級的分析師和資料科學家更輕鬆地使用機器學習技術。您可以將想要訓練模型的資料，以及與資料輸入相關聯的中繼資料提供給 Amazon Redshift。然後，Amazon Redshift ML 會建立可擷取輸入資料中模式的模型。然後，您可以使用這些模型來產生新輸入資料的預測，而且不會產生額外費用。

Amazon Redshift ML 如何與 Amazon 一起工作 SageMaker

Amazon Redshift 與 Amazon 自動 SageMaker 駕駛儀合作，以自動獲得最佳模型，並在 Amazon Redshift 中使用預測功能。

下圖說明Amazon Redshift ML 的運作方式。



一般工作流程如下：

1. Amazon Redshift會 將訓練資料匯出至 Amazon S3。
2. Amazon SageMaker 自動輔助駕駛會預先處理訓練資料。預處理會執行重要功能，例如輸入遺漏值。其會識別特定資料欄是可分類的 (例如郵遞區號)，正確格式化這些資料欄以進行訓練，並執行許多其他工作。選擇要套用於訓練資料集的最佳預處理器本身就是個問題，而 Amazon SageMaker Autopilot 會自動執行解決方案。
3. Amazon SageMaker Autopilot 會尋找演算法和演算法超參數，以最準確的預測提供模型。
4. Amazon Redshift 會將預測函數註冊為您 Amazon Redshift 叢集中的 SQL 函數。
5. 當您執行建立模型陳述式時，Amazon Redshift 會使用 Amazon SageMaker 進行訓練。因此，訓練模型會產生相關的成本。這是您 AWS 帳單 SageMaker 中 Amazon 的單獨商品項目。您也需要支付

Amazon S3 用於存放訓練資料的儲存費用。使用透過 CREATE MODEL 建立且可在 Redshift 叢集上編譯和執行的模型進行推論不收費。使用 Amazon Redshift ML 不會收取額外的 Amazon Redshift 費用。

主題

機器學習概述

透過使用 Amazon Redshift ML，您可以使用 SQL 陳述式訓練機器學習模型，並在 SQL 查詢中調用它們以進行預測。

若要協助您學習如何使用 Amazon Redshift ML，您可以觀看下列影片：Amazon Redshift ML。

若要了解設定 Redshift 叢集、許可和擁有權以使用 Amazon Redshift ML 的先決條件，請閱讀以下各節。這些章節也會說明簡單訓練和預測在 Amazon Redshift ML 中的運作方式。

機器學習如何解決問題

機器學習模型會在訓練資料中尋找模式，然後將這些模式套用至新資料，以產生預測。在機器學習中，您可以透過學習最能解釋您資料的模式來訓練這些模型。然後，您可以使用模型對新資料進行預測 (也稱為推論)。機器學習通常是一個反覆的過程，您可以透過變更參數和改進訓練資料來繼續提高預測的準確性。如果資料變更，則會使用新資料集重新訓練新模型。

為了解決各種業務目標，有不同的基本機器學習方法可用。

在 Amazon Redshift ML 中進行監督式學習

Amazon Redshift 支援受監督式學習，這是進階企業分析最常用的方法。當您擁有一組已建立的資料並了解特定輸入資料如何預測各種業務成果時，監督式學習是首選的機器學習方法。這些結果有時稱為標籤。特別是，如果您的資料集是一個包含特徵 (輸入) 和目標 (輸出) 的屬性資料表。例如，假設您有一個提供過去和現在客戶的年齡和郵遞區號的資料表。假設您還有一個「作用中」欄位，而此欄位對於現在的客戶而言是 true，對於已暫停其會員資格的客戶來說是 false。監督式機器學習的目標是發現導致

客戶流失的年齡和郵遞區號模式，以其目標為 “False” 的客戶表示。您可以使用此模型來預測可能流失的客戶，例如暫停其會員資格，並可能提供保留獎勵。

Amazon Redshift 支援監督式學習，其中包括迴歸、二進制分類和多類別分類。迴歸是指預測連續值的問題，例如客戶的總支出。二進制分類指的是預測兩個結果之一的問題，例如預測客戶是否會流失。多類別分類是指預測許多結果之一的問題，例如預測客戶可能感興趣的項目。資料分析師和資料科學家可以使用它來執行監督式學習，以解決諸如預測、個性化或客戶流失預測等問題。您還可以在諸如預測哪些銷售會結束、收入預測、詐騙偵測和客戶終身價值預測等問題中使用監督式學習。

在 Amazon Redshift ML 中進行非監督式學習

非監督式學習會使用機器學習演算法來分析和分組未標籤的訓練資料。演算法會發現隱藏的模式或分組。目標是為資料中的基礎結構或分佈建立模型，以進一步了解資料。

Amazon Redshift 支援 K 平均值叢集演算法，以解決非監督式學習問題。此演算法可解決您想在資料中探索分組的叢集問題。K 平均值演算法會嘗試尋找資料中的離散群組。未分類的資料會根據其相似性和差異進行分組和分割。透過分組，K 平均值演算法會反覆決定最佳質心，並將每個成員指派給最接近的質心。最接近屬於同一群組的相同質心的成員。群組的成員會盡可能地與相同群組中的其他成員相似，並盡可能地和其他群組的成員不同。例如，K 平均值叢集演算法可用於對受到流行病影響的城市進行分類，或根據消費品產品的受歡迎程度對城市進行分類。

使用 K 平均值演算法時，您可以指定輸入 k，指定要在資料中尋找的叢集數目。該演算法的輸出是 k 個質心的集合。每個資料點都屬於 k 個叢集中最接近它的其中一個叢集。每個叢集都會由其質心描述。質心可以被認為是叢集的多維平均值。K 平均值演算法會比較距離，以了解叢集彼此之間有多不同。較大的距離通常表示叢集之間的差異較大。

預處理資料對 K 平均值來說很重要，因為可確保模型的特徵保持相同的比例並產生可靠的結果。Amazon Redshift 支持一些 K 均值預處理器的創建模型語句，例如 StandardScaler， MinMax 和。 NumericPassthrough如果您不想對 K-means 應用任何預處理，請 NumericPassthrough 明確選擇作為變壓器。如需 K 平均值的相關資訊，請參閱 CREATE MODEL 與 K-MEANS 參數。

若要協助您了解如何使用 K 平均值叢集執行非監督式訓練，您可以觀看以下影片：使用 K 平均值叢集進行非監督式訓練。

Amazon Redshift ML 的術語和概念

下列術語用來描述 Amazon Redshift ML 的一些概念。

- Amazon Redshift 中的機器學習會使用一個 SQL 命令來訓練模型。Amazon Redshift ML 和 Amazon SageMaker 管理適當模型的所有數據轉換，許可，資源使用和發現。

- 訓練是 Amazon Redshift 透過在模型中執行指定的資料子集來建立機器學習模型的階段。Amazon Redshift 會自動在 Amazon 啟動培訓任務 SageMaker 並生成一個模型。
- 預測 (也稱為推論) 是在 Amazon Redshift SQL 查詢中使用模型來預測結果。在推論時，Amazon Redshift 會使用以模型為基礎的預測函數做為較大查詢的一部分來產生預測。預測是在 Redshift 叢集本機上計算的，因此可提供高輸送量、低延遲和零額外成本。
- 透過使用自己的模型 (BYOM)，您可以在 Amazon Redshift 以外的地方使用經過訓練的模型搭配 Amazon Redshift 進行本機 SageMaker 資料庫內推論。Amazon Redshift ML 支援使用 BYOM 進行本機推論。
- 當模型在 Amazon 預先訓練 SageMaker、由 Amazon SageMaker Neo 編譯並在亞馬 Amazon Redshift ML 中進行本地化時，會使用本地推論。若要將支援本機推論的模型匯入 Amazon Redshift，請使用 CREATE MODEL 命令。Amazon Redshift 通過調用 Amazon SageMaker 新導入預先訓練的 SageMaker 模型。您可以在那裡編譯模型，然後將編譯後的模型匯入 Amazon Redshift。使用本機推論以加快速度並降低成本。
- 當 Amazon Redshift 叫用部署於中的模型端點時，會使用遠端推論。 SageMaker遠端推論提供了調用所有類型的自訂模型和深度學習模型的靈活性，例如您在 Amazon SageMaker 中建立和部署的 TensorFlow 模型。

同樣重要的是以下各項：

- Amazon SageMaker 是全受管的機器學習服務。透過 Amazon SageMaker，資料科學家和開發人員可以輕鬆地在生產就緒的託管環境中建置、訓練和直接部署模型。有關 Amazon 的信息 SageMaker，請參閱 Amazon SageMaker 開發人員指南 SageMaker中的 Amazon 是什麼。
- Amazon SageMaker Autopilot 是一種功能集，可根據您的資料自動訓練和調整最佳機器學習模型以進行分類或回歸。您可以保持完整的控制權和可見性。Amazon SageMaker 自動輔助駕駛支援表格格式的輸入資料。Amazon SageMaker Autopilot 提供自動資料清理和預處理、線性迴歸、二進位分類和多類別分類的自動演算法選擇。也支援自動超參數最佳化 (HPO)、分散式訓練、自動執行個體和叢集大小選擇。如需 Amazon SageMaker 自動輔助駕駛儀的相關資訊，請參閱 Amazon SageMaker 開發人員指南中的使用 Amazon SageMaker Autopilot 自動駕駛儀自動開發模

適用於新手和專家的機器學習

Amazon Redshift ML 可讓您使用單一 SQL CREATE MODEL 命令來訓練模型。CREATE MODEL 命令會建立一個模型，讓 Amazon Redshift 使用此模型來產生具有熟悉 SQL 建構模組的模型式預測。

當您沒有機器學習、工具、語言、演算法和 API 的專業知識時，Amazon Redshift ML 會特別有用。使用 Amazon Redshift ML 時，您不必執行與外部機器學習服務整合所需的無差別繁重工作。Amazon

Redshift 可為您節省格式化和移動資料、管理許可控制或建置自訂整合、工作流程和指令碼的時間。您可以輕鬆使用熱門的機器學習演算法，並簡化從訓練到預測期間頻繁迭代的訓練需求。Amazon Redshift 會自動探索最佳演算法，並針對您的問題調整最佳模型。您可以從 Amazon Redshift 叢集中進行預測，而不需要將資料移出 Amazon Redshift，也不需要與其他服務互動並支付費用。

Amazon Redshift ML 可支援資料分析師和資料科學家使用機器學習。這也讓機器學習專家可以利用他們的知識來引導 CREATE MODEL 陳述式僅使用他們指定的層面。透過這樣做，您可以加快 CREATE MODEL 需要找到最佳候選項目的時間，提高模型的準確性，或兩者兼具。

CREATE MODEL 陳述式可在如何指定訓練工作的參數上提供彈性。使用此彈性，機器學習新手或專家就可以選擇自己喜歡的預處理器、演算法、問題類型和超參數。例如，對客戶流失感興趣的使用者可能會針對 CREATE MODEL 陳述式指定問題類型為二進制分類，而這非常適用於客戶流失。然後，CREATE MODEL 陳述式會將最佳模型的搜尋範圍縮小為二進制分類模型。即使使用者選擇了問題類型，CREATE MODEL 陳述式仍然可以使用許多選項。例如，CREATE MODEL 會探索並套用最佳的預處理轉換，並探索最佳的超參數設定。

Amazon Redshift ML 透過使用 Amazon 自動駕駛 SageMaker儀自動尋找最佳模型，讓訓練變得更加輕鬆。在幕後，Amazon SageMaker Autopilot 會根據您提供的資料自動訓練和調整最佳機器學習模型。然後，Amazon SageMaker Neo 會編譯訓練模型，並使其可用於在 Redshift 叢集中進行預測。當您使用訓練的模型執行機器學習推論查詢時，查詢可以使用 Amazon Redshift 的大量平行處理功能。同時，查詢可以使用以機器學習為基礎的預測。

- 身為機器學習初學者，您可能對於機器學習的不同層面有大致了解，例如預處理器、演算法和超參數，請只針對您指定的層面使用 CREATE MODEL 陳述式。然後您可以縮短 CREATE MODEL 需要找到最佳候選項目的時間，或是提高模型的準確性。此外，您可以透過引入其他領域知識 (例如問題類型或目標)，來增加預測的商業價值。例如，在客戶流失案例中，如果「客戶不活躍」的結果很少見，則 F1 目標通常高於準確度目標。因為高準確度模型可能會一直預測到「客戶很活躍」，此結果雖然具有高準確度，但商業價值很小。如需 F1 目標的相關資訊，請參閱 Amazon SageMaker API 參考JobObjective中的 AutoML。

  如需 CREATE MODEL 陳述式之基本選項的相關資訊，請參閱 簡易 CREATE MODEL。
- 若身為機器學習進階從業人員，您可以為某些 (但不是全部) 特徵指定問題類型和預處理器。然後，CREATE MODEL 會遵循您在指定層面上提供的建議。同時，CREATE MODEL 仍會發現用於其餘特徵的最佳的預處理器和最佳超參數。如需如何限制訓練管道的一或多個層面的相關資訊，請參閱 CREATE MODEL 和使用者指引。
- 若是身為機器學習專家，您可以完全掌控訓練和超參數調整。然後 CREATE MODEL 陳述式就不會嘗試探索最佳的預處理器、演算法和超參數，因為您做出了所有選擇。如需如何搭配 AUTO OFF 使用 CREATE MODEL 的相關資訊，請參閱 CREATE XGBoost 模型與 AUTO OFF。

- 身為資料工程師，您可以在亞馬遜中引入預先訓練的 XGBoost 模型， SageMaker 並將其匯入 Amazon Redshift 以進行本機推論。透過使用自己的模型 (BYOM)，您可以在 Amazon Redshift 以外的地方使用經過訓練的模型搭配 Amazon Redshift 進行本機 SageMaker資料庫內推論。Amazon Redshift ML 支援使用 BYOM 進行本機或遠端推論。

  如需如何針對本機或遠端推論使用 CREATE MODEL 陳述式的相關資訊，請參閱 使用自有模型 (BYOM) - 本機推論。

身為 Amazon Redshift ML 使用者，您可以選擇下列任何選項來訓練和部署模型：

- 問題類型，請參閱 CREATE MODEL 和使用者指引。
- 目標，請參閱 CREATE MODEL 和使用者指引 或 CREATE XGBoost 模型與 AUTO OFF。
- 模型類型，請參閱 CREATE XGBoost 模型與 AUTO OFF。
- 預處理器，請參閱 CREATE MODEL 和使用者指引。
- 超參數，請參閱 CREATE XGBoost 模型與 AUTO OFF。
- 使用自有模型 (BYOM)，請參閱 使用自有模型 (BYOM) - 本機推論。

使用 Amazon Redshift ML 的成本

Amazon Redshift ML 會使用您現有的叢集資源進行預測，因此您可以避免額外的 Amazon Redshift 費用。建立或使用模型不會產生額外的 Amazon Redshift 費用。預測發生在 Redshift 叢集本機，因此除非您需要調整叢集大小，否則您無需支付額外費用。Amazon Redshift ML 使用 Amazon SageMaker 來訓練您的模型，這確實會產生額外的相關費用。

Amazon Redshift 叢集中執行的預測函數不需額外費用。創建模型語句使用 Amazon， SageMaker 並產生額外的費用。成本會隨著訓練資料中的儲存格數量而增加。儲存格數量是記錄數目 (訓練查詢或資料表中) 乘以欄數的乘積。例如，當 CREATE MODEL 陳述式的 SELECT 查詢建立 10,000 條記錄和 5 個資料欄時，其建立的儲存格數量就是 50,000。

在某些情況下，由 CREATE MODEL 的 SELECT 查詢生成的訓練資料會超過您提供的 MAX_CELLS 限制 (若沒有提供限制，則是預設的 100 萬)。在這些情況下，CREATE MODEL 會隨機選擇大約的 MAX_CELLS (即訓練資料集中的「資料欄數目」記錄)。CREATE MODEL 會接著使用這些隨機選擇的元組執行訓練。隨機抽樣可確保減少的訓練資料集沒有任何偏差。因此，透過設定 MAX_CELLS，您可以控制訓練成本。

使用 CREATE MODEL 陳述式時，您可以使用 MAX_CELLS 和 MAX_RUNTIME 選項來控制成本、時間和潛在模型準確度。

MAX_RUNTIME 會指定使用「自動開啟」或「關閉」選項時，訓練可以採取的最大時間量。SageMaker 視資料集大小而定，訓練工作通常會比 MAX_RUNTIME 更快完成。模型訓練完成後，Amazon Redshift 會在背景執行其他工作，以便在叢集中編譯和安裝模型。因此，CREATE MODEL 可能需要比 MAX_RUNTIME 更長的時間才能完成。但是，MAX_RUNTIME 會限制用於訓練模型的計算量和時間。 SageMaker 您可以隨時使用 SHOW MODEL 來檢查模型的狀態。

當您在自動開啟的情況下執行建立模型時，Amazon Redshift ML 會使用 SageMaker 自動輔助駕駛功能自動智慧地探索不同的模型 (或候選模型)，以找到最佳模型。MAX_RUNTIME 會限制花費的時間和計算量。如果 MAX_RUNTIME 設太低，則可能甚至沒有足夠時間來探索一個候選項目。如果您看到錯誤「Autopilot 候選項目沒有模型」，請以較大的 MAX_RUNTIME 值重新執行 CREATE MODEL。如需有關此參數的詳細資訊，請參閱 Amazon SageMaker API 參考JobRuntimeInSeconds中的 MaxAutoML。

當您在自動關閉的情況下執行建立模型時，MAX_RUNTIME 會對應於訓練工作執行的時間長度限制。 SageMaker根據資料集的大小和使用的其他參數 (例如 MODEL_TYPE XGBOOST 中的 num_rounds)，訓練工作通常會更快完成。

您也可以在執行 CREATE MODEL 時指定較小的 MAX_CELLS 值，以控制成本或縮短訓練時間。儲存格是資料庫中的項目。每一列會對應至與資料欄數量一樣的儲存格，可以是固定寬度或不同寬度。MAX_CELLS 會限制儲存格的數量，因此也會限制用於訓練模型的訓練範例數。依預設，MAX_CELLS 會設定為 100 萬個儲存格。減少 MAX_CELL 會減少 Amazon Redshift 匯出並傳送給訓練模型的建立模型中 SELECT 查詢結果的 SageMaker 列數。因此，減少 MAX_CELLS 會減少使用 AUTO ON 和 AUTO OFF 來訓練模型的資料集大小。此方法有助於減少訓練模型的成本和時間。若要查看特定訓練任務的訓練和計費時間的相關資訊，請選擇 Amazon 中的訓練任務 SageMaker。

增加 MAX_RUNTIME 和 MAX_CELS 通常允許探索更多候選項 SageMaker 來改善模型品質。這樣，SageMaker 可能需要更多時間來訓練每個候選人，並使用更多數據來訓練更好的模型。如果您想要加快資料集的迭代或探索，請使用較低的 MAX_RUNTIME 和 MAX_CELLS。如果您想要改善模型的精確度，請使用較高的 MAX_RUNTIME 和 MAX_CELLS。

如需各種儲存格數目的成本和免費試用的相關資訊，請參閱 Amazon Redshift 定價。

開始使用 Amazon RedShift ML

Amazon Redshift ML 可讓 SQL 使用者使用熟悉的 SQL 命令輕鬆建立、訓練和部署機器學習模型。使用 Amazon Redshift ML，您可以使用 Redshift 叢集中的資料，透過 Amazon 訓練模型。 SageMaker 之後，這些模型會經過本地化，並且可以在 Amazon Redshift 資料庫中進行預測。Amazon Redshift ML 目前支援機器學習演算法 XGBoost (AUTO ON 和 OFF) 和多層感知器 (AUTO ON)、K 平均值 (AUTO OFF) 和線性學習程式。

主題

針對 Amazon Redshift ML 管理進行叢集和組態設定

在您使用 Amazon Redshift ML 之前，請先完成叢集設定並設定使用 Amazon Redshift ML 的許可。

使用 Amazon Redshift ML 的叢集設定

使用 Amazon Redshift ML 之前，請先完成以下先決條件。

身為 Amazon Redshift 管理員，請執行下列一次性設定。

若要為 Amazon Redshift ML 執行一次性叢集設定

1. 使用 AWS Management Console 或 AWS Command Line Interface (AWS CLI) 建立 Redshift 叢集。建立叢集時，請務必附加 AWS Identity and Access Management (IAM) 政策。如需將 Amazon Redshift ML 與 Amazon 搭配使用所需許可的詳細資訊 SageMaker，請參閱將 Amazon Redshift 機器學習 (ML) 與亞馬遜搭配使用所需的許可。 SageMaker
2. 使用下列其中一個方法，建立使用 Amazon Redshift ML 所需的 IAM 角色：

   - 其中一個簡單的操作是搭配 `AmazonS3FullAccess` 和 `AmazonSageMakerFullAccess` 政策建立一個 IAM 角色，以與 Amazon Redshift ML 搭配使用。如果您打算同時建立預測模型，也可以將 `AmazonForecastFullAccess` 政策附加至您的角色。
   - 我們建議您透過 Amazon Redshift 主控台建立 IAM 角色，其具有有權執行 SQL 命令 (例如 CREATE MODEL) 的 `AmazonRedshiftAllCommandsFullAccess` 政策。Amazon Redshift 使用順暢的 API 型機制，以程式設計方式代表您 AWS 帳戶 建立 IAM 角色。Amazon Redshift 會自動將現有的 AWS 受管政策附加到 IAM 角色。此方法意味著您可以保留在 Amazon Redshift 主控台內，而不必切換到 IAM 主控台來建立角色。如需詳細資訊，請參閱建立 IAM 角色做為 Amazon Redshift 的預設值。

     當 IAM 角色建立為叢集的預設值時，請加入 `redshift` 做為資源名稱的一部分，或使用 RedShift 專屬標籤來標記這些資源。

如果您的叢集已開啟增強型 Amazon VPC 路由，您可以使用透過 Amazon Redshift 主控台建立的 IAM 角色。此 IAM 角色已附加 AmazonRedshiftAllCommandsFullAccess 政策，並將以下許可新增到政策中。這些額外許可會允許 Amazon Redshift 建立和刪除您帳戶中的彈性網絡介面 (ENI)，並將其附加到 Amazon EC2 或 Amazon ECS 上執行的編譯工作。如此一來，Amazon S3 儲存貯體中的物件只能從虛擬私有雲端 (VPC) 內存取，而網際網路存取會遭到封鎖。

```
{
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterfacePermission",
        "ec2:DeleteNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:CreateNetworkInterface",
        "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": "*"
}
```

- 如果您想要建立具有更嚴格政策的 IAM 角色，您可以使用以下政策。您也可以修改此政策以因應您的需求。

  Amazon S3 儲存貯體 redshift-downloads/redshift-ml/ 是用於其他步驟和範例之範例資料的儲存所在位置。如果您不需要從 Amazon S3 載入資料，則可以將其移除。或者，也可以將其取代為您用來將資料載入 Amazon Redshift 的其他 Amazon S3 儲存貯體。

  your-account-id、*your-role* 和 *your-s3-bucket* 值是您在 CREATE MODEL 命令中指定的值。

  (選擇性) 如果您在使用 Amazon Redshift ML 時指定 AWS KMS 金鑰，請使用 AWS KMS 範例政策的金鑰區段。*your-kms-key* 值是您用來作為 CREATE MODEL 命令一部分的金鑰。

  ```
  {
      "Version": "2012-10-17",
  ```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetAuthorizationToken",
                "ecr:GetDownloadUrlForLayer",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "sagemaker:*Job*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "s3:AbortMultipartUpload",
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:iam::<your-account-id>:role/<your-role>",
                "arn:aws:s3:::<your-s3-bucket>/*",
                "arn:aws:s3:::redshift-downloads/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<your-s3-bucket>",
                "arn:aws:s3:::redshift-downloads"
            ]
        }
        // Optional section needed if you use AWS KMS keys.
```

```
        ,{
            "Effect": "Allow",
            "Action": [
                "kms:CreateGrant",
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:Encrypt",
                "kms:GenerateDataKey*"
             ],
             "Resource": [
                "arn:aws:kms:<your-region>:<your-account-id>:key/<your-kms-key>"
             ]
        }
    ]
}
```

3. 若要允許 Amazon Redshift 並 SageMaker 假設該角色與其他服務互動，請將下列信任政策新增至 IAM 角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "redshift.amazonaws.com",
          "sagemaker.amazonaws.com",
          "forecast.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. (選擇性) 建立 Amazon S3 儲存貯體和 AWS KMS 金鑰。這些是供 Amazon Redshift 用於存儲發送到 Amazon 的培訓數據， SageMaker 並從 Amazon SageMaker 接收訓練過的模型。
5. (選擇性) 建立 IAM 角色和 Amazon S3 儲存貯體的不同組合，以控制對不同使用者群組的存取。
6. (選擇性) 為 Redshift 叢集開啟 VPC 人雲端路由時，請為 Redshift 叢集所在的 VPC 人雲 SageMaker 端建立 Amazon S3 端點和端點。這樣做可在 CREATE MODEL 期間，讓服務之間的

流量通過您的 VPC。如需 VPC 路由的相關資訊，請參閱 Amazon Redshift 中的增強型 VPC 路由。

如需為超參數調整任務指定私有 VPC 所需許可的詳細資訊，請參閱將 Amazon Redshift ML 與 Amazon SageMaker 搭配使用所需的許可。

如需有關如何使用 CREATE MODEL 陳述式開始針對不同使用案例建立模型的資訊，請參閱 CREATE MODEL。

管理許可和擁有權

就像其他資料庫物件 (例如資料表或函數) 一樣，Amazon Redshift 會繫結建立和使用機器學習模型來存取控制機制。建立執行預測函數的模型有不同的許可。

下列範例使用兩個使用者群組 `retention_analyst_grp` (模型建立者) 和 `marketing_analyst_grp` (模型使用者) 來說明 Amazon Redshift 如何管理存取控制。保留分析師會建立機器學習模型，讓其他一組使用者可透過取得的許可使用這些模型。

超級使用者可以使用下列陳述式 GRANT (授予) USER 或 GROUP 許可，以建立機器學習模型。

```
GRANT CREATE MODEL TO GROUP retention_analyst_grp;
```

如果使用者擁有 SCHEMA 的一般 CREATE 許可，則具有此許可的使用者或群組可以在叢集中的任何結構描述中建立模型。機器學習模型是結構描述階層的一部分，與資料表、檢視表、程序和使用者定義函數類似。

假設結構描述 `demo_ml` 已存在，請依照下列方式授予兩個使用者群組結構描述上的許可。

```
GRANT CREATE, USAGE ON SCHEMA demo_ml TO GROUP retention_analyst_grp;
```

```
GRANT USAGE ON SCHEMA demo_ml TO GROUP marketing_analyst_grp;
```

若要讓其他使用者使用您的機器學習推論功能，請授予 EXECUTE 權限。下列範例會使用 EXECUTE 權限，將使用模型的許可授予 marketing_analyst_grp GROUP。

```
GRANT EXECUTE ON MODEL demo_ml.customer_churn_auto_model TO GROUP
 marketing_analyst_grp;
```

搭配 CREATE MODEL 和 EXECUTE 使用 REVERSE 陳述式，可撤銷使用者或群組的這些許可。如需許可控制命令的相關資訊，請參閱 GRANT 和 REVOKE。

使用模型可解釋性與 Amazon Redshift ML

透過 Amazon Redshift ML 中的模型可解釋性，您可以使用特徵重要性值，協助您了解訓練資料中的每個屬性如何影響預測結果。

模型可解釋性可以解釋模型進行的預測，以協助改善您的機器學習 (ML) 模型。模型可解釋性可幫助解釋這些模型如何使用特徵歸因方法進行預測。

Amazon Redshift ML 整合了模型可解釋性，為 Amazon Redshift ML 使用者提供模型解釋功能。如需有關模型解釋性的詳細資訊，請參閱什麼是 Machine Learning 預測的公平性和模型解釋能力？ 在 Amazon 開 SageMaker 發人員指南。

模型解釋性還可監控模型在生產環境中為特徵屬性漂移所做的推論。它還提供了一些工具來幫助您生成模型治理報告，您可以使用這些報告告知風險和合規團隊以及外部監管機構。

當您在使用 CREATE MODEL 陳述式時指定「自動開啟」或「自動關閉」選項時，模型訓練工作完成後， SageMaker 會建立說明輸出。您可以使用 EXPLAIIN_MODEL 函數來查詢 JSON 格式的解釋性報告。如需詳細資訊，請參閱 機器學習函數。

Amazon Redshift ML 概率指標

在監督式學習問題中，類別標籤是使用輸入資料的預測結果。例如，如果您使用模型來預測客戶是否會重新訂閱串流服務，則可能的標籤是很可能和不太可能。Redshift ML 會提供概率指標的功能，可為每個標籤指派概率以指出其可能性。這可以幫助您根據預測的結果做出更明智的決策。在 Amazon Redshift ML 中，建立具有二進制分類或多類別分類問題類型的 AUTO ON 模型時，都可以使用概率指標。如果您省略 AUTO ON 參數，Redshift ML 會假設模型應具有 AUTO ON。

建立模型

建立模型時，Amazon Redshift 會自動偵測模型類型和問題類型。如果是分類問題，Redshift 會自動建立第二個推論函數，您可以使用該函數輸出相對於每個標籤的機率。第二個推論函數的名稱是您指定的推論函數名稱接著字串 `_probabilities`。例如，如果您將推論函數命名為 `customer_churn_predict`，則第二個推論函數的名稱為 `customer_churn_predict_probabilities`。然後，您可以查詢此函數以獲取每個標籤的概率。

```
CREATE MODEL customer_churn_model
FROM customer_activity
```

```
    PROBLEM_TYPE BINARY_CLASSIFICATION
TARGET churn
FUNCTION customer_churn_predict
IAM_ROLE {default}
AUTO ON
SETTINGS ( S3_BUCKET '<DOC-EXAMPLE-BUCKET>'
```

獲取概率

一旦概率函數準備就緒，執行命令會傳回一個 SUPER 類型，其中包含傳回的概率陣列及其相關標籤。例如，結果 `"probabilities" : [0.7, 0.3], "labels" : ["False.", "True."]` 表示 False 標籤的概率為 0.7，而 True 標籤的概率為 0.3。

```
SELECT customer_churn_predict_probabilities(Account_length, Area_code,
            VMail_message, Day_mins, Day_calls, Day_charge,Eve_mins, Eve_calls,
            Eve_charge, Night_mins, Night_calls, Night_charge,Intl_mins, Intl_calls,
            Intl_charge, Cust_serv_calls)
FROM customer_activity;

customer_churn_predict_probabilities
 --------------------
 {"probabilities" : [0.7, 0.3], "labels" : ["False.", "True."]}
 {"probabilities" : [0.8, 0.2], "labels" : ["False.", "True."]}
 {"probabilities" : [0.75, 0.25], "labels" : ["True.", "False"]}
```

概率和標籤陣列一律會按其概率降序排序。您可以撰寫查詢，只傳回具有最高概率的預測標籤，方法是將 SUPER 傳回概率函數結果解除巢狀結構。

```
SELECT prediction.labels[0], prediction.probabilities[0]
            FROM (SELECT customer_churn_predict_probabilities(Account_length,
 Area_code,
            VMail_message, Day_mins, Day_calls, Day_charge,Eve_mins, Eve_calls,
            Eve_charge, Night_mins, Night_calls, Night_charge,Intl_mins, Intl_calls,
            Intl_charge, Cust_serv_calls) AS prediction
FROM customer_activity);

  labels   | probabilities
-----------+--------------
 "False."  | 0.7
 "False."  | 0.8
 "True."   | 0.75
```

為了使查詢更簡單，您可以將預測函數的結果儲存在資料表中。

```
CREATE TABLE churn_auto_predict_probabilities AS
             (SELECT customer_churn_predict_probabilities(Account_length, Area_code,
             VMail_message, Day_mins, Day_calls, Day_charge,Eve_mins, Eve_calls,
             Eve_charge, Night_mins, Night_calls, Night_charge,Intl_mins,
             Intl_calls, Intl_charge, Cust_serv_calls) AS prediction
FROM customer_activity);
```

您可以查詢含有結果的資料表，以僅傳回概率高於 0.7 的預測。

```
SELECT prediction.labels[0], prediction.probabilities[0]
FROM churn_auto_predict_probabilities
WHERE prediction.probabilities[0] > 0.7;

  labels   | probabilities
-----------+--------------
 "False."  | 0.8
 "True."   | 0.75
```

使用索引符號，你可以得到特定標籤的概率。下列範例會傳回所有 `True.` 標籤的概率。

```
SELECT label, index, p.prediction.probabilities[index]
FROM churn_auto_predict_probabilities p, p.prediction.labels AS label AT index
WHERE label='True.';

  label  | index | probabilities
---------+-------+---------------
 "True." |     0 | 0.3
 "True." |     0 | 0.2
 "True." |     0 | 0.75
```

下列範例會傳回所有具有 `True.` 標籤且概率大於 0.7 的資料列，這表示客戶可能流失。

```
SELECT prediction.labels[0], prediction.probabilities[0]
FROM churn_auto_predict_probabilities
WHERE prediction.probabilities[0] > 0.7 AND prediction.labels[0] = "True.";

labels     | probabilities
-----------+--------------
 "True."   | 0.75
```

Amazon Redshift ML 的教學課程

您可以透過 Amazon Redshift ML，使用 SQL 陳述式訓練機器學習模型，然後在 SQL 查詢中調用這些模型以進行預測。Amazon Redshift 中的機器學習會使用一個 SQL 命令來訓練模型。Amazon Redshift 會自動在 Amazon 啟動培訓任務 SageMaker 並生成一個模型。建立模型之後，您可以使用模型的預測函數在 Amazon Redshift 中執行預測。

請遵循這些教學課程中的步驟以瞭解 Amazon Redshift ML 特徵：

- 教學課程：建置客戶流失模型
- 教學課程：建置遠端推論模型
- 教學課程：建置 K 平均值叢集模型
- 教學課程：建置多類別分類模型
- 教學課程：建置 XGBoost 模型
- 教學課程：建置迴歸模型
- 教學課程：使用線性學習程式建置迴歸模型
- 教學課程：使用線性學習程式建置多類別分類模型

教學課程：建置客戶流失模型

在本教學課程中，您可以使用 Amazon Redshift ML 透過 CREATE MODEL 命令建立客戶流失模型，並針對使用者案例執行預測查詢。然後，您可以使用 CREATE MODEL 命令所產生的 SQL 函數來實作查詢。

您可以使用簡單的 CREATE MODEL 命令來匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATE MODEL 陳述式，將訓練資料指定為資料表或 SELECT 陳述式。

此範例會使用歷史資訊來建構行動電信業者的客戶流失機器學習模型。首先， SageMaker 訓練您的機器學習模型，然後使用任意客戶的設定檔資訊測試您的模型。驗證模型後，Amazon SageMaker 將模型和預測功能部署到 Amazon Redshift。您可以使用預測函數來預測客戶是否會流失。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他二進制分類問題，例如預測銷售潛在客戶是否會結案。您還可以預測金融交易是否為詐騙。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

若要完成本教學課程，您需要以下先決條件：

- 您必須為 Amazon Redshift ML 設定一個 Amazon Redshift 叢集。若要這麼做，請使用針對 Amazon Redshift ML 管理進行叢集和組態設定的文件。
- 您用來建立模型的 Amazon Redshift 叢集，以及用來暫存訓練資料和儲存模型成品的 Amazon S3 儲存貯體必須位於相同 AWS 區域。
- 若要下載 SQL 命令和本文件中使用的範例資料集，請執行下列其中一項：
  - 下載 SQL 命令、客戶活動檔案和鮑魚檔案。
  - 使用對 AWS CLI 於 Amazon S3，運行以下命令。您可以使用自己的目標路徑。

```
aws s3 cp s3://redshift-downloads/redshift-ml/tutorial-scripts/redshift-ml-
tutorial.sql </target/path>
aws s3 cp s3://redshift-downloads/redshift-ml/customer_activity/
customer_activity.csv </target/path>
aws s3 cp s3://redshift-downloads/redshift-ml/abalone_xgb/abalone_xgb.csv </target/
path>
```

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 Amazon Redshift 查詢編輯器 v2 編輯和執行查詢，以及視覺化結果。

執行下列查詢會建立名為 `customer_activity` 的資料表，並從 Amazon S3 擷取範例資料集。

```
DROP TABLE IF EXISTS customer_activity;

CREATE TABLE customer_activity (
state varchar(2),
account_length int,
area_code int,
phone varchar(8),
intl_plan varchar(3),
```

```
vMail_plan varchar(3),
vMail_message int,
day_mins float,
day_calls int,
day_charge float,
total_charge float,
eve_mins float,
eve_calls int,
eve_charge float,
night_mins float,
night_calls int,
night_charge float,
intl_mins float,
intl_calls int,
intl_charge float,
cust_serv_calls int,
churn varchar(6),
record_date date
);

COPY customer_activity
FROM 's3://redshift-downloads/redshift-ml/customer_activity/'
REGION 'us-east-1' IAM_ROLE default
FORMAT AS CSV IGNOREHEADER 1;
```

步驟 2：建立機器學習模型

流失率是我們在此模型中的目標輸入。模型的所有其他輸入都是有助於建立函數以預測流失的屬性。

下列範例會使用 CREATE MODEL 作業，並使用客戶年齡、郵遞區號、支出和案例等輸入，來提供預測客戶是否處於作用中狀態的模型。在下列範例中，請將 *DOC-EXAMPLE-BUCKET* 取代為您的 Amazon S3 儲存貯體。

```
CREATE MODEL customer_churn_auto_model
FROM
    (
      SELECT state,
             account_length,
             area_code,
             total_charge/account_length AS average_daily_spend,
             cust_serv_calls/account_length AS average_daily_cases,
             churn
      FROM customer_activity
```

```
      WHERE  record_date < '2020-01-01'
     )
TARGET churn FUNCTION ml_fn_customer_churn_auto
IAM_ROLE default SETTINGS (
  S3_BUCKET '<DOC-EXAMPLE-BUCKET>'
);
```

上述範例中的 SELECT 查詢會建立訓練資料。TARGET 子句會指定 CREATE MODEL 操作用來學習如何預測的機器學習標籤資料欄。目標資料欄「流失率」會指出客戶是否仍然具有作用中的會員資格或已暫停會員資格。S3_BUCKET 欄位是您先前建立的 Amazon S3 儲存貯體名稱。Amazon S3 存儲桶用於在 Amazon 紅移和亞馬遜之間共享培訓數據和成品。 SageMaker其餘的資料欄是用於預測的特徵。

如需 CREATE MODEL 命令的基本使用案例的語法和特徵摘要，請參閱簡單 CREATE MODEL。

新增伺服器端加密的許可 (選用)

Amazon Redshift 默認情況下使用 Amazon SageMaker 自動駕駛儀進行培訓。特別是，Amazon Redshift 會將訓練資料安全地匯出到客戶指定的 Amazon S3 儲存貯體。如果您未指定KMS_KEY_ID，資料會預設為使用伺服器端加密 SSE-S3 進行加密。

當您使用具有 AWS KMS 受管理金鑰 (SSE-MMS) 的伺服器端加密輸入時，請新增下列權限：

```
{
    "Effect": "Allow",
    "Action": [
    "kms:Encrypt"
    "kms:Decrypt"
    ]
}
```

如需 Amazon SageMaker 角色的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的 Amazon SageMaker 角色。

檢查模型訓練的狀態 (選擇性)

您可以使用 SHOW MODEL 命令來知道模型何時準備就緒。

使用下列操作來檢查模型的狀態。

```
SHOW MODEL customer_churn_auto_model;
```

下列為上一個操作的輸出範例。

```
+--------------------------
+-----------------------------------------------------------------------------------------------------------------
+
|           Key            |
                 Value
             |
+--------------------------
+-----------------------------------------------------------------------------------------------------------------
+
|        Model Name        |
        customer_churn_auto_model
             |
|       Schema Name        |
                 public
             |
|          Owner           |
                 awsuser
             |
|      Creation Time       |
        Tue, 14.06.2022 17:15:52
             |
|       Model State        |
                TRAINING
             |
|                          |

             |
|      TRAINING DATA:      |

             |
|          Query           | SELECT STATE, ACCOUNT_LENGTH, AREA_CODE, TOTAL_CHARGE /
 ACCOUNT_LENGTH AS AVERAGE_DAILY_SPEND, CUST_SERV_CALLS / ACCOUNT_LENGTH AS
 AVERAGE_DAILY_CASES, CHURN |
|                          |
         FROM CUSTOMER_ACTIVITY
             |
|                          |
    WHERE RECORD_DATE < '2020-01-01'
             |
|      Target Column       |
                 CHURN
             |
```

```
|                          |

             |
|       PARAMETERS:        |

             |
|        Model Type        |
                   auto
             |
|       Problem Type       |

             |
|        Objective         |

             |
|     AutoML Job Name      |
     redshiftml-20220614171552640901
             |
|      Function Name       |
        ml_fn_customer_churn_auto
             |
|   Function Parameters    |                                                  state
 account_length area_code average_daily_spend average_daily_cases
                        |
| Function Parameter Types |
      varchar int4 int4 float8 int4
             |
|         IAM Role         |
          default-aws-iam-role
             |
|        S3 Bucket         |
             DOC-EXAMPLE-BUCKET
             |
|       Max Runtime        |
                   5400
             |
+--------------------------
+-----------------------------------------------------------------------------------------------------------------
+
```

模型訓練完成後，model_state 變數會變成 Model is Ready，且預測函數會變為可用。

步驟 3：執行模型的預測

您可以使用 SQL 陳述式來檢視預測模型所做的預測。在此範例中，CREATE MODEL 操作所建立的預測函數會命名為 `ml_fn_customer_churn_auto`。預測函數的輸入引數會對應於特徵的類型，例如 varchar 會用於 `state`，而整數會用於 `account_length`。預測函數的輸出類型與 CREATE MODEL 陳述式的 TARGET 資料欄相同。

1. 您用了 2020-01-01 之前的資料訓練模型，因此現在您可以在測試集上使用預測功能。以下查詢顯示 2020-01-01 之後註冊的客戶是否會經歷流失的預測。

```
SELECT
    phone,
    ml_fn_customer_churn_auto(
        state,
        account_length,
        area_code,
        total_charge / account_length,
        cust_serv_calls / account_length
    ) AS active
FROM
    customer_activity
WHERE
    record_date > '2020-01-01';
```

2. 下列範例針對不同的使用案例使用相同的預測函數。在此案例中，Amazon Redshift 會從不同州的客戶中 (記錄日期大於 2020-01-01)，預測流失者和非流失者的比例。

```
WITH predicted AS (
    SELECT
        state,
        ml_fn_customer_churn_auto(
            state,
            account_length,
            area_code,
            total_charge / account_length,
            cust_serv_calls / account_length
        ) :: varchar(6) AS active
    FROM
        customer_activity
    WHERE
        record_date > '2020-01-01'
)
```

```
SELECT
    state,
    SUM(
        CASE
            WHEN active = 'True.' THEN 1
            ELSE 0
        END
    ) AS churners,
    SUM(
        CASE
            WHEN active = 'False.' THEN 1
            ELSE 0
        END
    ) AS nonchurners,
    COUNT(*) AS total_per_state
FROM
    predicted
GROUP BY
    state
ORDER BY
    state;
```

3. 下列範例會使用預測函數來預測某州中流失的客戶百分比。在此案例中，Amazon Redshift 會預測記錄日期大於 2020-01-01 的流失百分比。

```
WITH predicted AS (
    SELECT
        state,
        ml_fn_customer_churn_auto(
            state,
            account_length,
            area_code,
            total_charge / account_length,
            cust_serv_calls / account_length
        ) :: varchar(6) AS active
    FROM
        customer_activity
    WHERE
        record_date > '2020-01-01'
)
SELECT
    state,
    CAST((CAST((SUM(
```

```
        CASE
            WHEN active = 'True.' THEN 1
            ELSE 0
        END
    )) AS FLOAT) / CAST(COUNT(*) AS FLOAT)) AS DECIMAL (3, 2)) AS pct_churn,
    COUNT(*) AS total_customers_per_state
FROM
    predicted
GROUP BY
    state
ORDER BY
    3 DESC;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon RedshiftML 的成本
- CREATE MODEL 命令
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

教學課程：建置遠端推論模型

以下教學將介紹如何建立隨機切割森林模型的步驟 SageMaker，該模型先前已在 Amazon Redshift 之外進行訓練和部署。隨機分割森林演算法會偵測資料集中的異常資料點。使用遠端推論建立模型可讓您將隨機切割森林 SageMaker 模型帶入 Amazon Redshift。然後，在 Amazon Redshift 中，您可以使用 SQL 在遠 SageMaker 端端點上執行預測。

您可以使用「建立模型」命令從 Amazon SageMaker 端點匯入機器學習模型，並準備 Amazon Redshift 預測功能。使用 CREATE MODEL 作業時，您需要提供 SageMaker 機器學習模型的端點名稱。

在本教學中，您會使用模型端點建立 Amazon Redshift 機器學習 SageMaker 模型。一旦您的機器學習模型準備就緒，您就可以使用它在 Amazon Redshift 中執行預測。首先，您訓練並在 Amazon 中建立端點 SageMaker，然後取得端點名稱。然後，您可以使用 CREATE MODEL 命令建立具有 Amazon Redshift ML 的模型。最後，您可以使用 CREATE MODEL 指令產生的預測函數對模型執行預測。

使用案例範例

您可以使用隨機分割森林模型和遠端推論來偵測其他資料集中的異常狀況，例如預測電子商務交易的快速增加或減少。您還可以預測天氣或地震活動的顯著變化。

工作

- 必要條件
- 步驟 1：部署 Amazon SageMaker 模型
- 步驟 2：取得模 SageMaker 型端點
- 步驟 3：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 4：建立 Amazon Redshift ML 的模型
- 步驟 5：執行模型的預測

必要條件

若要完成本教學課程，您需要以下先決條件：

- 您已完成 Amazon Redshift ML 的管理設定。
- 您已下載紐約市計程車資料集、建立 Amazon S3 儲存貯體，然後將資料上傳到 Amazon S3 儲存貯體。
- 您必須訓練、部署 SageMaker 模型和端點，並取得 SageMaker 端點的名稱。使用此 AWS CloudFormation 範本可自動佈建 AWS 帳戶中的所有 SageMaker 資源。

步驟 1：部署 Amazon SageMaker 模型

1. 若要部署模型，請移至 Amazon SageMaker 主控台，在導覽窗格中的 [筆記本] 下選擇 [筆記本] 執行個體。
2. 為範本所建立的 Jupyter 記事本選擇「開啟 Jupyter」。 CloudFormation
3. 選擇 `bring-your-own-model-remote-inference.ipynb`。
4. 將以下幾行取代為 Amazon S3 儲存貯體和字首，設定參數來將訓練輸入和輸出儲存在 Amazon S3 中。

```
data_location=f"s3://{bucket}/{prefix}/",
output_path=f"s3://{bucket}/{prefix}/output",
```

5. 選擇快轉按鈕以執行所有儲存格。

步驟 2：取得模 SageMaker 型端點

在 Amazon 主 SageMaker 控台的導覽窗格中的推論下，選擇端點並尋找您的型號名稱。在 Amazon Redshift 中建立遠端推論模型時，必須複製模型的端點名稱。

步驟 3：將資料從 Amazon S3 載入到 Amazon Redshift

使用 Amazon Redshift 查詢編輯器 v2 在 Amazon Redshift 中執行以下 SQL 命令。如果 `rcf_taxi_data` 資料表存在，這些命令會捨棄資料表、建立相同名稱的資料表，然後將範例資料集載入資料表中。

```
DROP TABLE IF EXISTS public.rcf_taxi_data CASCADE;

CREATE TABLE public.rcf_taxi_data (ride_timestamp timestamp, nbr_passengers int);

COPY public.rcf_taxi_data
FROM
    's3://sagemaker-sample-files/datasets/tabular/anomaly_benchmark_taxi/
NAB_nyc_taxi.csv'
    IAM_ROLE default
    IGNOREHEADER 1
    FORMAT AS CSV;
```

步驟 4：建立 Amazon Redshift ML 的模型

執行下列查詢，以使用您在上一個步驟中取得的 SageMaker 模型端點在 Amazon Redshift ML 中建立模型。*randomcutforest-xxxxxxxxx*用您自己的 SageMaker 端點名稱替換。

```
CREATE MODEL public.remote_random_cut_forest
FUNCTION remote_fn_rcf(int)
RETURNS decimal(10, 6) SAGEMAKER '<randomcutforest-xxxxxxxxx>' IAM_ROLE default;
```

檢查模型狀態 (選擇性)

您可以使用 SHOW MODEL 命令來知道模型何時準備就緒。

使用 SHOW MODEL 來檢查模型的狀態。

```
SHOW MODEL public.remote_random_cut_forest
```

輸出顯示 SageMaker 端點和函數名稱。

```
+-------------------------+----------------------------------------+
|        Model Name       |       remote_random_cut_forest         |
+-------------------------+----------------------------------------+
|       Schema Name       |                 public                 |
|          Owner          |                awsuser                 |
|      Creation Time      |       Wed, 15.06.2022 17:58:21         |
|       Model State       |                 READY                  |
|                         |                                        |
|       PARAMETERS:       |                                        |
|         Endpoint        |     <randomcutforest-xxxxxxxxx>        |
|      Function Name      |              remote_fn_rcf             |
|      Inference Type     |                 Remote                 |
| Function Parameter Types |                 int4                  |
|         IAM Role        |         default-aws-iam-role           |
+-------------------------+----------------------------------------+
```

步驟 5：執行模型的預測

Amazon SageMaker 隨機切割森林演算法旨在偵測資料集中的異常資料點。在此範例中，您的模型旨在偵測由於重要事件而導致的計程車乘車尖峰。您可以透過為每個資料點產生異常分數，使用模型來預測異常事件。

使用下列查詢來計算整個計程車資料集的異常分數。請注意，您會參考上一個步驟中在 CREATE MODEL 陳述式中使用的函數。

```
SELECT
    ride_timestamp,
    nbr_passengers,
    public.remote_fn_rcf(nbr_passengers) AS score
FROM
    public.rcf_taxi_data;
```

檢查是高分和低分的異常 (選擇性)

執行下列查詢，以尋找分數大於平均分數三個標準差的任何資料點。

```
WITH score_cutoff AS (
    SELECT
        STDDEV(public.remote_fn_rcf(nbr_passengers)) AS std,
        AVG(public.remote_fn_rcf(nbr_passengers)) AS mean,
        (mean + 3 * std) AS score_cutoff_value
    FROM
        public.rcf_taxi_data
)
SELECT
    ride_timestamp,
    nbr_passengers,
    public.remote_fn_rcf(nbr_passengers) AS score
FROM
    public.rcf_taxi_data
WHERE
    score > (
        SELECT
            score_cutoff_value
        FROM
            score_cutoff
    )
ORDER BY
    2 DESC;
```

執行下列查詢，以尋找分數大於平均分數三個標準差的任何資料點。

```
WITH score_cutoff AS (
    SELECT
        STDDEV(public.remote_fn_rcf(nbr_passengers)) AS std,
        AVG(public.remote_fn_rcf(nbr_passengers)) AS mean,
        (mean - 3 * std) AS score_cutoff_value
    FROM
        public.rcf_taxi_data
)
SELECT
    ride_timestamp,
    nbr_passengers,
    public.remote_fn_rcf(nbr_passengers) AS score
FROM
    public.rcf_taxi_data
WHERE
    score < (
        SELECT
```

```
            score_cutoff_value
        FROM
            score_cutoff
    )
ORDER BY
    2 DESC;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon Redshift ML 的成本
- CREATE MODEL 操作
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

教學課程：建置 K 平均值叢集模型

在本教學課程中，您會使用 Amazon Redshift ML 來建立、訓練和部署以 K 平均值演算法為基礎的機器學習模型。此演算法可解決您想在資料中探索分組的叢集問題。K 平均值有助於對尚未標記的資料進行分組。若要進一步了解 K 均值叢集，請參閱 Amazon 開發人員指南中的 K 均值叢集如何運作。SageMaker

您將使用 CREATE MODEL 操作，從 Amazon Redshift 叢集建立 K 平均值模型。您可以使用 CREATE MODEL 命令來匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATE MODEL 操作，將訓練資料指定為資料表或 SELECT 陳述式。

在本教學課程中，您會在全球事件、語言和音調資料庫 (Global Database of Events, Language, and Tone (GDELT)) 資料集上使用 K 平均值，此資料集會監控世界各地的新聞，並每天每秒地儲存這些資料。K 平均值會將音調、動作者或位置相似的事件進行分組。資料會以多個檔案形式儲存在 Amazon Simple Storage Service 中 (儲存在兩個不同的資料夾中)。這些資料夾為歷史資料夾 (涵蓋 1979–2013 年) 及每日更新資料夾 (涵蓋 2013 年及以後的年份)。在此範例中，我們使用歷史格式，並帶入 1979 年的資料。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他叢集問題，例如對在串流服務上具有類似檢視習慣的客戶進行分組。您也可以使用 Redshift ML 來預測遞送服務的最佳託運中心數量。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的管理設定。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

1. 使用 Amazon Redshift 查詢編輯器 v2 來執行下列查詢。查詢會在公用結構描述中捨棄 gdelt_data 資料表 (如果資料表存在)，並在公用結構描述中建立相同名稱的資料表。

```
DROP TABLE IF EXISTS gdelt_data CASCADE;

CREATE TABLE gdelt_data (
    GlobalEventId bigint,
    SqlDate bigint,
    MonthYear bigint,
    Year bigint,
    FractionDate double precision,
    Actor1Code varchar(256),
    Actor1Name varchar(256),
    Actor1CountryCode varchar(256),
    Actor1KnownGroupCode varchar(256),
    Actor1EthnicCode varchar(256),
    Actor1Religion1Code varchar(256),
    Actor1Religion2Code varchar(256),
    Actor1Type1Code varchar(256),
    Actor1Type2Code varchar(256),
    Actor1Type3Code varchar(256),
    Actor2Code varchar(256),
    Actor2Name varchar(256),
    Actor2CountryCode varchar(256),
```

```
    Actor2KnownGroupCode varchar(256),
    Actor2EthnicCode varchar(256),
    Actor2Religion1Code varchar(256),
    Actor2Religion2Code varchar(256),
    Actor2Type1Code varchar(256),
    Actor2Type2Code varchar(256),
    Actor2Type3Code varchar(256),
    IsRootEvent bigint,
    EventCode bigint,
    EventBaseCode bigint,
    EventRootCode bigint,
    QuadClass bigint,
    GoldsteinScale double precision,
    NumMentions bigint,
    NumSources bigint,
    NumArticles bigint,
    AvgTone double precision,
    Actor1Geo_Type bigint,
    Actor1Geo_FullName varchar(256),
    Actor1Geo_CountryCode varchar(256),
    Actor1Geo_ADM1Code varchar(256),
    Actor1Geo_Lat double precision,
    Actor1Geo_Long double precision,
    Actor1Geo_FeatureID bigint,
    Actor2Geo_Type bigint,
    Actor2Geo_FullName varchar(256),
    Actor2Geo_CountryCode varchar(256),
    Actor2Geo_ADM1Code varchar(256),
    Actor2Geo_Lat double precision,
    Actor2Geo_Long double precision,
    Actor2Geo_FeatureID bigint,
    ActionGeo_Type bigint,
    ActionGeo_FullName varchar(256),
    ActionGeo_CountryCode varchar(256),
    ActionGeo_ADM1Code varchar(256),
    ActionGeo_Lat double precision,
    ActionGeo_Long double precision,
    ActionGeo_FeatureID bigint,
    DATEADDED bigint
);
```

2. 下列查詢會將範例資料載入 gdelt_data 資料表中。

```
COPY gdelt_data
```

```
FROM 's3://gdelt-open-data/events/1979.csv'
REGION 'us-east-1'
IAM_ROLE default
CSV
DELIMITER '\t';
```

檢查訓練資料 (選擇性)

若要查看您的模型將在哪些資料上進行訓練，請使用下列查詢。

```
SELECT
    AvgTone,
    EventCode,
    NumArticles,
    Actor1Geo_Lat,
    Actor1Geo_Long,
    Actor2Geo_Lat,
    Actor2Geo_Long
FROM
    gdelt_data LIMIT 100;
```

步驟 2：建立機器學習模型

下列範例使用 CREATE MODEL 命令來建立可將資料分組為七個叢集的模型。K 值是資料點將分散在其中的叢集數目。模型會將您的資料點分類成叢集，其中資料點彼此更相似。透過將資料點分成群組，K 平均值演算法會反覆判斷最佳的叢集中心。然後演算法會將每個資料點指派給最近的叢集中心。最接近屬於同一群組的相同叢集中心的成員。群組的成員會盡可能地與相同群組中的其他成員相似，並盡可能地和其他群組的成員不同。K 值是主觀的，取決於測量資料點之間相似性的方法。如果叢集分佈不均，您可以變更 K 值以平滑化叢集大小。

在下列範例中，請將 *DOC-EXAMPLE-BUCKET* 取代為您的 Amazon S3 儲存貯體。

```
CREATE MODEL news_data_clusters
FROM
    (
        SELECT
            AvgTone,
            EventCode,
            NumArticles,
            Actor1Geo_Lat,
            Actor1Geo_Long,
```

```
            Actor2Geo_Lat,
            Actor2Geo_Long
        FROM
            gdelt_data
    ) FUNCTION news_monitoring_cluster
    IAM_ROLE default
    AUTO OFF
    MODEL_TYPE KMEANS
    PREPROCESSORS 'none'
    HYPERPARAMETERS DEFAULT
    EXCEPT
    (K '7')
    SETTINGS (S3_BUCKET '<DOC-EXAMPLE-BUCKET>');
```

檢查模型訓練的狀態 (選擇性)

您可以使用 SHOW MODEL 命令來知道模型何時準備就緒。

若要檢查模型狀態，請使用下列 SHOW MODEL 操作並找到 `Model State` 是否為 `Ready`。

```
SHOW MODEL NEWS_DATA_CLUSTERS;
```

當模型準備就緒時，先前操作的輸出應會顯示 `Model State` 為 `Ready`。下列為 SHOW MODEL 操作的輸出範例。

```
+--------------------------
+-----------------------------------------------------------------------------------------------------------
+
|        Model Name        |
 news_data_clusters                                    |
+--------------------------
+-----------------------------------------------------------------------------------------------------------
+
|       Schema Name        |                                                    public
                                          |
|          Owner           |                                                  awsuser
                                          |
|      Creation Time       |                                      Fri, 17.06.2022
 16:32:19                                      |
|       Model State        |                                                   READY
                                          |
|        train:msd         |                                           2973.822754
                                          |
```

```
|      train:progress      |                                                  100.000000
                                        |
|     train:throughput     |                                               237114.875000
                                        |
|      Estimated Cost      |                                                  0.004983
                                        |
|                          |
                                        |
|      TRAINING DATA:      |
                                        |
|          Query           | SELECT AVGTONE, EVENTCODE, NUMARTICLES, ACTOR1GEO_LAT,
 ACTOR1GEO_LONG, ACTOR2GEO_LAT, ACTOR2GEO_LONG |
|                          |                                                FROM GDELT_DATA
                                        |
|                          |
                                        |
|       PARAMETERS:        |
                                        |
|        Model Type        |                                                     kmeans
                                        |
|    Training Job Name     |
 redshiftml-20220617163219978978-kmeans                                  |
|      Function Name       |
 news_monitoring_cluster                                        |
|   Function Parameters    |       avgtone eventcode numarticles actor1geo_lat
 actor1geo_long actor2geo_lat actor2geo_long        |
| Function Parameter Types |                              float8 int8 int8 float8 float8
 float8 float8                          |
|         IAM Role         |                                        default-aws-iam-
role                                      |
|        S3 Bucket         |                                           DOC-EXAMPLE-
BUCKET                                     |
|       Max Runtime        |                                                5400
                                        |
|                          |
                                        |
|     HYPERPARAMETERS:     |
                                        |
|       feature_dim        |                                                  7
                                        |
|            k             |                                                  7
                                        |
```

```
+--------------------------
+-------------------------------------------------------------------------------------------------------------------
+
```

步驟 3：執行模型的預測

識別叢集

您可以找到模型在資料中識別的離散群組 (也稱為叢集)。比起任何其他叢集中心，叢集是更接近其叢集中心的一組資料點。由於 K 值代表模型中的叢集數目，因此也代表叢集中心的數目。下面的查詢會透過顯示與每個 globaleventid 相關聯的叢集來識別叢集。

```
SELECT
    globaleventid,
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS cluster
FROM
    gdelt_data;
```

檢查資料的分佈

您可以檢查叢集之間的資料分佈，以查看您選擇的 K 值是否會讓資料稍微均勻地分佈。使用下列查詢來判斷資料是否平均分佈在叢集中。

```
SELECT
    events_cluster,
    COUNT(*) AS nbr_events
FROM
    (
        SELECT
            globaleventid,
            news_monitoring_cluster(
                AvgTone,
                EventCode,
                NumArticles,
```

```
                Actor1Geo_Lat,
                Actor1Geo_Long,
                Actor2Geo_Lat,
                Actor2Geo_Long
            ) AS events_cluster
        FROM
            gdelt_data
    )
GROUP BY
    1;
```

請注意，如果叢集分佈不均，您可以變更 K 值以平滑化叢集大小。

判斷叢集中心

比起任何其他叢集中心，資料點會更接近其叢集中心。因此，尋找叢集中心可協助您定義叢集。

執行下列查詢，根據事件代碼的文章數目判斷叢集的中心。

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    eventcode,
    SUM(numArticles) AS numArticles
FROM
    gdelt_data
GROUP BY
    1,
    2;
```

顯示叢集中資料點的相關資訊

使用以下查詢傳回指派給第五個叢集的點的資料。選定的文章必須有兩個動作者。

```
SELECT
    news_monitoring_cluster (
```

```
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    eventcode,
    actor1name,
    actor2name,
    SUM(numarticles) AS totalarticles
FROM
    gdelt_data
WHERE
    events_cluster = 5
    AND actor1name <> ' '
    AND actor2name <> ' '
GROUP BY
    1,
    2,
    3,
    4
ORDER BY
    5 desc;
```

顯示具有相同族裔代碼的動作者的事件資料

下列查詢會計算以正面語調撰寫有關事件的文章數目。該查詢也會要求兩個動作者具有相同的族裔代碼，並傳回每個事件分派給哪個叢集。

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    SUM(numarticles) AS total_articles,
    eventcode AS event_code,
```

```
    Actor1EthnicCode AS ethnic_code
FROM
    gdelt_data
WHERE
    Actor1EthnicCode = Actor2EthnicCode
    AND Actor1EthnicCode <> ' '
    AND Actor2EthnicCode <> ' '
    AND AvgTone > 0
GROUP BY
    1,
    3,
    4
HAVING
    (total_articles) > 4
ORDER BY
    1,
    2 ASC;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon Redshift ML 的成本
- CREATE MODEL 操作
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

教學課程：建置多類別分類模型

在本教學課程中，您會使用 Amazon Redshift ML 建立可解決多類別分類問題的機器學習模型。多類別分類演算法將資料點分類為三個或多個類別之一。然後，您可以使用 CREATE MODEL 命令所產生的 SQL 函數來實作查詢。

您可以使用 CREATE MODEL 命令來匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATE MODEL 操作，將訓練資料指定為資料表或 SELECT 陳述式。

若要按照本教學課程進行操作，您可以使用公共資料集電子商務銷售預測，其中包括線上英國零售商的銷售資料。您產生的模型將以特殊客戶忠誠度計劃中最活躍的客戶為目標。透過多類別分類，您可以使用該模型來預測客戶在 13 個月之間有多少個月是活躍的。預測函數會指定預計會活躍 7 個月或更長時間的客戶來加入該計劃。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他多類別分類問題，例如從產品線預測暢銷產品。您還可以預測圖像包含哪些水果，例如選擇蘋果、梨子或柳橙。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的管理設定。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 Amazon Redshift 查詢編輯器 v2 來執行下列查詢。這些查詢會將範例資料載入 Amazon Redshift。

1. 以下範例會查詢名為 ecommerce_sales 的資料表。

```
CREATE TABLE IF NOT EXISTS ecommerce_sales (
    invoiceno VARCHAR(30),
    stockcode VARCHAR(30),
    description VARCHAR(60),
    quantity DOUBLE PRECISION,
    invoicedate VARCHAR(30),
    unitprice DOUBLE PRECISION,
    customerid BIGINT,
    country VARCHAR(25)
);
```

2. 下列查詢會將電子商務銷售預測資料集中的範例資料複製到 ecommerce_sales 資料表中。

```
COPY ecommerce_sales
FROM
    's3://redshift-ml-multiclass/ecommerce_data.txt'
IAM_ROLE default
DELIMITER '\t'
IGNOREHEADER 1
REGION 'us-east-1'
MAXERROR 100;
```

分割資料

當您在 Amazon Redshift ML 中建立模型時， SageMaker 會自動將資料分割為訓練和測試集， SageMaker 以便判斷模型的準確性。透過在此步驟手動分割資料，您將能夠透過配置額外預測集來驗證模型的準確性。

使用下列 SQL 陳述式將資料分割成三組，以進行訓練、驗證和預測。

```
--creates table with all data
CREATE TABLE ecommerce_sales_data AS (
    SELECT
        t1.stockcode,
        t1.description,
        t1.invoicedate,
        t1.customerid,
        t1.country,
        t1.sales_amt,
        CAST(RANDOM() * 100 AS INT) AS data_group_id
    FROM
        (
            SELECT
                stockcode,
                description,
                invoicedate,
                customerid,
                country,
                SUM(quantity * unitprice) AS sales_amt
            FROM
                ecommerce_sales
            GROUP BY
                1,
                2,
```

```
                3,
                4,
                5
        ) t1
);

--creates training set
CREATE TABLE ecommerce_sales_training AS (
    SELECT
        a.customerid,
        a.country,
        a.stockcode,
        a.description,
        a.invoicedate,
        a.sales_amt,
        (b.nbr_months_active) AS nbr_months_active
    FROM
        ecommerce_sales_data a
        INNER JOIN (
            SELECT
                customerid,
                COUNT(
                    DISTINCT(
                        DATE_PART(y, CAST(invoicedate AS DATE)) || '-' || LPAD(
                            DATE_PART(mon, CAST(invoicedate AS DATE)),
                            2,
                            '00'
                        )
                    )
                ) AS nbr_months_active
            FROM
                ecommerce_sales_data
            GROUP BY
                1
        ) b ON a.customerid = b.customerid
    WHERE
        a.data_group_id < 80
);

--creates validation set
CREATE TABLE ecommerce_sales_validation AS (
    SELECT
        a.customerid,
        a.country,
```

```
        a.stockcode,
        a.description,
        a.invoicedate,
        a.sales_amt,
        (b.nbr_months_active) AS nbr_months_active
    FROM
        ecommerce_sales_data a
        INNER JOIN (
            SELECT
                customerid,
                COUNT(
                    DISTINCT(
                        DATE_PART(y, CAST(invoicedate AS DATE)) || '-' || LPAD(
                            DATE_PART(mon, CAST(invoicedate AS DATE)),
                            2,
                            '00'
                        )
                    )
                ) AS nbr_months_active
            FROM
                ecommerce_sales_data
            GROUP BY
                1
        ) b ON a.customerid = b.customerid
    WHERE
        a.data_group_id BETWEEN 80
        AND 90
);

--creates prediction set
CREATE TABLE ecommerce_sales_prediction AS (
    SELECT
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    FROM
        ecommerce_sales_data
    WHERE
        data_group_id > 90);
```

步驟 2：建立機器學習模型

在此步驟中，您可以使用 CREATE MODEL 陳述式，使用多類別分類來建立機器學習模型。

下列查詢會使用 CREATE MODEL 操作建立具有訓練集的多類別分類模型。將 *DOC-EXAMPLE-BUCKET* 取代為您自己的 Amazon S3 儲存貯體。

```
CREATE MODEL ecommerce_customer_activity
FROM
    (
        SELECT
            customerid,
            country,
            stockcode,
            description,
            invoicedate,
            sales_amt,
            nbr_months_active
        FROM
            ecommerce_sales_training
    ) TARGET nbr_months_active FUNCTION predict_customer_activity IAM_ROLE default
 PROBLEM_TYPE MULTICLASS_CLASSIFICATION SETTINGS (
        S3_BUCKET '<DOC-EXAMPLE-BUCKET>',
        S3_GARBAGE_COLLECT OFF
    );
```

在此查詢中，您可以將問題類型指定為 `Multiclass_Classification`。您為模型預測的目標是 `nbr_months_active`。 SageMaker 完成模型訓練後，它會建立函數`predict_customer_activity`，您將使用該函數在 Amazon Redshift 中進行預測。

顯示模型訓練的狀態 (選擇性)

您可以使用 SHOW MODEL 命令來知道模型何時準備就緒。

使用下列查詢傳回模型的各種指標，包括模型狀態和準確度。

```
SHOW MODEL ecommerce_customer_activity;
```

當模型準備就緒時，先前操作的輸出應會顯示 `Model State` 為 `Ready`。下列為 SHOW MODEL 操作的輸出範例。

```
+--------------------------
+-----------------------------------------------------------------------------------------------------
+
|        Model Name        |
 ecommerce_customer_activity                                |
+--------------------------
+-----------------------------------------------------------------------------------------------------
+
|       Schema Name        |                                                 public
                                  |
|          Owner           |                                                 awsuser
                                  |
|      Creation Time       |                                  Fri, 17.06.2022 19:02:15
                                  |
|       Model State        |                                                   READY
                                  |
|   Training Job Status    |
 MaxAutoMLJobRuntimeReached                                 |
|   validation:accuracy    |                                                0.991280
                                  |
|      Estimated Cost      |                                                7.897689
                                  |
|                          |
                                  |
|      TRAINING DATA:      |
                                  |
|          Query           | SELECT CUSTOMERID, COUNTRY, STOCKCODE, DESCRIPTION,
 INVOICEDATE, SALES_AMT, NBR_MONTHS_ACTIVE |
|                          |                                     FROM
 ECOMMERCE_SALES_TRAINING                                |
|      Target Column       |                                          NBR_MONTHS_ACTIVE
                                  |
|                          |
                                  |
|       PARAMETERS:        |
                                  |
|        Model Type        |                                                 xgboost
                                  |
|       Problem Type       |                                  MulticlassClassification
                                  |
|        Objective         |                                                Accuracy
                                  |
```

```
|     AutoML Job Name      |
 redshiftml-20220617190215268770                                |
|      Function Name       |
 predict_customer_activity                                      |
|   Function Parameters    |               customerid country stockcode description
 invoicedate sales_amt               |
| Function Parameter Types |                          int8 varchar varchar varchar
 varchar float8                        |
|         IAM Role         |                                    default-aws-iam-role
                                    |
|        S3 Bucket         |                                        DOC-EXAMPLE-BUCKET
                                    |
|       Max Runtime        |                                            5400
                                    |
+--------------------------
+-------------------------------------------------------------------------------------------------
+
```

步驟 3：執行模型的預測

以下查詢會顯示哪些客戶符合您客戶忠誠度計劃的資格。如果模型預測客戶至少有 7 個月的活躍時間，則模型會為忠誠度計劃選取該客戶。

```
SELECT
    customerid,
    predict_customer_activity(
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    ) AS predicted_months_active
FROM
    ecommerce_sales_prediction
WHERE
    predicted_months_active >= 7
GROUP BY
    1,
    2
LIMIT
    10;
```

針對驗證資料執行預測查詢 (選擇性)

針對驗證資料執行下列預測查詢，以查看模型的準確度層級。

```
SELECT
    CAST(SUM(t1.match) AS decimal(7, 2)) AS predicted_matches,
    CAST(SUM(t1.nonmatch) AS decimal(7, 2)) AS predicted_non_matches,
    CAST(SUM(t1.match + t1.nonmatch) AS decimal(7, 2)) AS total_predictions,
    predicted_matches / total_predictions AS pct_accuracy
FROM
    (
        SELECT
            customerid,
            country,
            stockcode,
            description,
            invoicedate,
            sales_amt,
            nbr_months_active,
            predict_customer_activity(
                customerid,
                country,
                stockcode,
                description,
                invoicedate,
                sales_amt
            ) AS predicted_months_active,
            CASE
                WHEN nbr_months_active = predicted_months_active THEN 1
                ELSE 0
            END AS match,
            CASE
                WHEN nbr_months_active <> predicted_months_active THEN 1
                ELSE 0
            END AS nonmatch
        FROM
            ecommerce_sales_validation
    )t1;
```

預測有多少客戶錯失加入機會 (選擇性)

下列查詢會比較預測只會在 5 或 6 個月內處於活躍狀態的客戶數量。該模型會預測這些客戶將錯過忠誠度計劃。然後，查詢會將幾乎不會錯過該計劃的數量與預測符合忠誠度計劃資格的數量進行比較。此

查詢可用於提供是否要降低忠誠度計劃門檻的決定。您還可以定判斷是否有大量的客戶預計幾乎不會錯過該計劃。然後，您可以鼓勵這些客戶增加他們的活動，以獲得忠誠度計劃會員資格。

```
SELECT
    predict_customer_activity(
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    ) AS predicted_months_active,
    COUNT(customerid)
FROM
    ecommerce_sales_prediction
WHERE
    predicted_months_active BETWEEN 5 AND 6
GROUP BY
    1
ORDER BY
    1 ASC
LIMIT
    10)
UNION
(SELECT
      NULL AS predicted_months_active,
    COUNT (customerid)
FROM
    ecommerce_sales_prediction
WHERE
    predict_customer_activity(
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    ) >=7);
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon Redshift ML 的成本
- CREATE MODEL 操作
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

教學課程：建置 XGBoost 模型

在本教學課程中，您會使用來自 Amazon S3 的資料建立模型，並使用 Amazon Redshift ML 透過該模型執行預測查詢。XGBoost 演算法是梯度提升樹演算法的最佳化實作。與其他梯度提升樹演算法相比，XGBoost 可處理更多的資料類型、關係和分佈。您可以將 XGBoost 用於迴歸、二進制分類、多類別分類和排名問題。如需 XGBoost 演算法的詳細資訊，請參閱 Amazon 開發人員指南中的 XGBoost 演算法。 SageMaker

Amazon Redshift ML `CREATE MODEL` 操作與 `AUTO OFF` 選項目前支援 XGBoost 作為 `MODEL_TYPE`。您可以根據您的使用案例提供相關資訊 (例如目標和超參數) 做為 `CREATE MODEL` 命令的一部分。

在本教學課程中，您將使用鈔票驗證資料集，這是一個二進制分類問題，用於預測給定鈔票是真的還是偽造的。

使用案例範例

您可以使用 Amazon Redshift ML 來解決其他二進制分類問題，例如預測病患是健康狀態還是患有疾病。您也可以預測電子郵件是否為垃圾郵件。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的管理設定。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 Amazon Redshift 查詢編輯器 v2 來執行下列查詢。

下列查詢會建立兩個資料表、從 Amazon S3 載入資料，然後將資料分割為訓練集和測試集。您將使用訓練集來訓練模型並建立預測函數。然後，您將在測試集上測試預測函數。

```
--create training set table
CREATE TABLE banknoteauthentication_train(
    variance FLOAT,
    skewness FLOAT,
    curtosis FLOAT,
    entropy FLOAT,
    class INT
);

--Load into training table
COPY banknoteauthentication_train
FROM
    's3://redshiftbucket-ml-sagemaker/banknote_authentication/train_data/' IAM_ROLE
 default REGION 'us-west-2' IGNOREHEADER 1 CSV;

--create testing set table
CREATE TABLE banknoteauthentication_test(
    variance FLOAT,
    skewness FLOAT,
    curtosis FLOAT,
    entropy FLOAT,
    class INT
);

--Load data into testing table
COPY banknoteauthentication_test
FROM
    's3://redshiftbucket-ml-sagemaker/banknote_authentication/test_data/'
    IAM_ROLE default
    REGION 'us-west-2'
    IGNOREHEADER 1
    CSV;
```

步驟 2：建立機器學習模型

下列查詢會透過上述步驟中建立的訓練集，在 Amazon Redshift ML 中建立 XGBoost 模型。將 DOC-EXAMPLE-BUCKET 取代為您自己的 S3_BUCKET，這將存儲您的輸入資料集和其他 Redshift ML 成品。

```
CREATE MODEL model_banknoteauthentication_xgboost_binary
FROM
    banknoteauthentication_train
    TARGET class
    FUNCTION func_model_banknoteauthentication_xgboost_binary
    IAM_ROLE default
    AUTO OFF
    MODEL_TYPE xgboost
    OBJECTIVE 'binary:logistic'
    PREPROCESSORS 'none'
    HYPERPARAMETERS DEFAULT
EXCEPT(NUM_ROUND '100')
SETTINGS(S3_BUCKET '<DOC-EXAMPLE-BUCKET>');
```

顯示模型訓練的狀態 (選擇性)

您可以使用 SHOW MODEL 命令來知道模型何時準備就緒。

使用下列查詢來監控模型訓練進度。

```
SHOW MODEL model_banknoteauthentication_xgboost_binary;
```

如果模型是 READY，則 SHOW MODEL 操作也會提供 train:error 指標，如下列輸出範例所示。train:error 指標用於測量模型準確度，可測量到小數點後六位。值為 0 是最準確的，而值為 1 是最不準確的。

```
+--------------------------+-------------------------------------------------+
|        Model Name        |   model_banknoteauthentication_xgboost_binary   |
+--------------------------+-------------------------------------------------+
| Schema Name              | public                                          |
| Owner                    | awsuser                                         |
| Creation Time            | Tue, 21.06.2022 19:07:35                        |
| Model State              | READY                                           |
| train:error              |                                        0.000000 |
| Estimated Cost           |                                        0.006197 |
|                          |                                                 |
```

```
| TRAINING DATA:           |                                                    |
| Query                    | SELECT *                                           |
|                          | FROM "BANKNOTEAUTHENTICATION_TRAIN"                |
| Target Column            | CLASS                                              |
|                          |                                                    |
| PARAMETERS:              |                                                    |
| Model Type               | xgboost                                            |
| Training Job Name        | redshiftml-20220621190735686935-xgboost            |
| Function Name            | func_model_banknoteauthentication_xgboost_binary |
| Function Parameters      | variance skewness curtosis entropy                 |
| Function Parameter Types | float8 float8 float8 float8                        |
| IAM Role                 | default-aws-iam-role                               |
| S3 Bucket                | DOC-EXAMPLE-BUCKET    |
| Max Runtime              |                                               5400 |
|                          |                                                    |
| HYPERPARAMETERS:         |                                                    |
| num_round                |                                                100 |
| objective                | binary:logistic                                    |
+--------------------------+----------------------------------------------------+
```

步驟 3：執行模型的預測

檢查模型的準確度

下列預測查詢會使用在上一個步驟中建立的預測函數來檢查模型的準確度。在測試集上執行此查詢，可確保模型與訓練集的對應不會過於緊密。這種緊密的對應關係也稱為過度擬合，過度擬合可能會導致模型做出不可靠的預測。

```
WITH predict_data AS (
    SELECT
        class AS label,
        func_model_banknoteauthentication_xgboost_binary (variance, skewness, curtosis,
 entropy) AS predicted,
        CASE
            WHEN label IS NULL THEN 0
            ELSE label
        END AS actual,
        CASE
            WHEN actual = predicted THEN 1 :: INT
            ELSE 0 :: INT
        END AS correct
```

```
    FROM
        banknoteauthentication_test
),
aggr_data AS (
    SELECT
        SUM(correct) AS num_correct,
        COUNT(*) AS total
    FROM
        predict_data
)
SELECT
    (num_correct :: FLOAT / total :: FLOAT) AS accuracy
FROM
    aggr_data;
```

預測真鈔和假鈔的數量

以下預測查詢會傳回測試集中真鈔和假鈔的預測數量。

```
WITH predict_data AS (
    SELECT
        func_model_banknoteauthentication_xgboost_binary(variance, skewness, curtosis,
 entropy) AS predicted
    FROM
        banknoteauthentication_test
)
SELECT
    CASE
        WHEN predicted = '0' THEN 'Original banknote'
        WHEN predicted = '1' THEN 'Counterfeit banknote'
        ELSE 'NA'
    END AS banknote_authentication,
    COUNT(1) AS count
FROM
    predict_data
GROUP BY
    1;
```

找到真鈔和假鈔的平均觀察值

下列預測查詢會傳回測試集中預測為真鈔和假鈔的每個鈔票特徵的平均值。

```
WITH predict_data AS (
```

```
    SELECT
        func_model_banknoteauthentication_xgboost_binary(variance, skewness, curtosis,
 entropy) AS predicted,
          variance,
          skewness,
          curtosis,
          entropy
    FROM
        banknoteauthentication_test
)
SELECT
    CASE
        WHEN predicted = '0' THEN 'Original banknote'
        WHEN predicted = '1' THEN 'Counterfeit banknote'
        ELSE 'NA'
    END AS banknote_authentication,
    TRUNC(AVG(variance), 2) AS avg_variance,
    TRUNC(AVG(skewness), 2) AS avg_skewness,
    TRUNC(AVG(curtosis), 2) AS avg_curtosis,
    TRUNC(AVG(entropy), 2) AS avg_entropy
FROM
    predict_data
GROUP BY
    1
ORDER BY
    2;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon Redshift ML 的成本
- CREATE MODEL 操作
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

教學課程：建置迴歸模型

在本教學中，您會使用 Amazon Redshift ML 建立機器學習迴歸模型，並在模型上執行預測查詢。迴歸模型允許您預測數值結果，例如房屋的價格，或有多少人將使用城市的自行車租賃服務。您可以在 Amazon Redshift 中使用 CREATE MODEL 命令搭配訓練資料。然後，Amazon Redshift ML 會編譯模型，將經過訓練的模型匯入 Redshift，並準備 SQL 預測函數。您可以在 Amazon Redshift 中的 SQL 查詢中使用預測函數。

在本教學課程中，您將使用 Amazon Redshift ML 建置迴歸模型，以預測在一天中的任何特定時間使用多倫多市自行車共享服務的人數。模型的輸入包括假日和天氣條件。您將使用迴歸模型，因為您想要此問題的數值結果。

您可以使用 CREATE MODEL 命令來匯出訓練資料、訓練模型、匯入模型，以及讓模型可在 Amazon Redshift 中作為 SQL 函數。使用 CREATE MODEL 操作，將訓練資料指定為資料表或 SELECT 陳述式。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他迴歸問題，例如預測客戶的終身價值。您也可以使用 Redshift ML 來預測最有利的價格和產生的產品收入。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：驗證模型

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的管理設定。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 Amazon Redshift 查詢編輯器 v2 來執行下列查詢。

1. 您必須建立三個資料表，才能將三個公開資料集載入 Amazon Redshift。資料集是多倫多自行車乘客資料、歷史氣象資料和歷史假期資料。在 Amazon Redshift 查詢編輯器中執行下列查詢，以建立名為 `ridership`、`weather` 和 `holiday` 的資料表。

```
CREATE TABLE IF NOT EXISTS ridership (
    trip_id INT,
    trip_duration_seconds INT,
    trip_start_time timestamp,
    trip_stop_time timestamp,
    from_station_name VARCHAR(50),
    to_station_name VARCHAR(50),
    from_station_id SMALLINT,
    to_station_id SMALLINT,
    user_type VARCHAR(20)
);

CREATE TABLE IF NOT EXISTS weather (
    longitude_x DECIMAL(5, 2),
    latitude_y DECIMAL(5, 2),
    station_name VARCHAR(20),
    climate_id BIGINT,
    datetime_utc TIMESTAMP,
    weather_year SMALLINT,
    weather_month SMALLINT,
    weather_day SMALLINT,
    time_utc VARCHAR(5),
    temp_c DECIMAL(5, 2),
    temp_flag VARCHAR(1),
    dew_point_temp_c DECIMAL(5, 2),
    dew_point_temp_flag VARCHAR(1),
    rel_hum SMALLINT,
    rel_hum_flag VARCHAR(1),
    precip_amount_mm DECIMAL(5, 2),
    precip_amount_flag VARCHAR(1),
    wind_dir_10s_deg VARCHAR(10),
    wind_dir_flag VARCHAR(1),
    wind_spd_kmh VARCHAR(10),
    wind_spd_flag VARCHAR(1),
    visibility_km VARCHAR(10),
    visibility_flag VARCHAR(1),
    stn_press_kpa DECIMAL(5, 2),
    stn_press_flag VARCHAR(1),
    hmdx SMALLINT,
    hmdx_flag VARCHAR(1),
    wind_chill VARCHAR(10),
    wind_chill_flag VARCHAR(1),
    weather VARCHAR(10)
```

```
);

CREATE TABLE IF NOT EXISTS holiday (holiday_date DATE, description VARCHAR(100));
```

2. 下列查詢會將範例資料載入您在上一個步驟中建立的資料表。

```
COPY ridership
FROM
    's3://redshift-ml-bikesharing-data/bike-sharing-data/ridership/'
    IAM_ROLE default
    FORMAT CSV
    IGNOREHEADER 1
    DATEFORMAT 'auto'
    TIMEFORMAT 'auto'
    REGION 'us-west-2'
    gzip;

COPY weather
FROM
    's3://redshift-ml-bikesharing-data/bike-sharing-data/weather/'
    IAM_ROLE default
    FORMAT csv
    IGNOREHEADER 1
    DATEFORMAT 'auto'
    TIMEFORMAT 'auto'
    REGION 'us-west-2'
    gzip;

COPY holiday
FROM
    's3://redshift-ml-bikesharing-data/bike-sharing-data/holiday/'
    IAM_ROLE default
    FORMAT csv
    IGNOREHEADER 1
    DATEFORMAT 'auto'
    TIMEFORMAT 'auto'
    REGION 'us-west-2'
    gzip;
```

3. 下列查詢會對 ridership 和 weather 資料集執行轉換，以移除偏差或異常。移除偏差和異常可改善模型準確度。該查詢會透過建立兩個名為 ridership_view 和 weather_view 的新檢視簡化資料表。

```
CREATE
OR REPLACE VIEW ridership_view AS
SELECT
    trip_time,
    trip_count,
    TO_CHAR(trip_time, 'hh24') :: INT trip_hour,
    TO_CHAR(trip_time, 'dd') :: INT trip_day,
    TO_CHAR(trip_time, 'mm') :: INT trip_month,
    TO_CHAR(trip_time, 'yy') :: INT trip_year,
    TO_CHAR(trip_time, 'q') :: INT trip_quarter,
    TO_CHAR(trip_time, 'w') :: INT trip_month_week,
    TO_CHAR(trip_time, 'd') :: INT trip_week_day
FROM
    (
        SELECT
            CASE
                WHEN TRUNC(r.trip_start_time) < '2017-07-01' :: DATE THEN
 CONVERT_TIMEZONE(
                    'US/Eastern',
                    DATE_TRUNC('hour', r.trip_start_time)
                )
                ELSE DATE_TRUNC('hour', r.trip_start_time)
            END trip_time,
            COUNT(1) trip_count
        FROM
            ridership r
        WHERE
            r.trip_duration_seconds BETWEEN 60
            AND 60 * 60 * 24
        GROUP BY
            1
    );

CREATE
OR REPLACE VIEW weather_view AS
SELECT
    CONVERT_TIMEZONE(
        'US/Eastern',
        DATE_TRUNC('hour', datetime_utc)
    ) daytime,
    ROUND(AVG(temp_c)) temp_c,
    ROUND(AVG(precip_amount_mm)) precip_amount_mm
FROM
```

```
    weather
GROUP BY
    1;
```

4. 下面的查詢會建立一個資料表，該資料表會將 ridership_view 和 weather_view 中的所有相關輸入屬性結合到 trip_data 資料表中。

```
CREATE TABLE trip_data AS
SELECT
    r.trip_time,
    r.trip_count,
    r.trip_hour,
    r.trip_day,
    r.trip_month,
    r.trip_year,
    r.trip_quarter,
    r.trip_month_week,
    r.trip_week_day,
    w.temp_c,
    w.precip_amount_mm,CASE
        WHEN h.holiday_date IS NOT NULL THEN 1
        WHEN TO_CHAR(r.trip_time, 'D') :: INT IN (1, 7) THEN 1
        ELSE 0
    END is_holiday,
    ROW_NUMBER() OVER (
        ORDER BY
            RANDOM()
    ) serial_number
FROM
    ridership_view r
    JOIN weather_view w ON (r.trip_time = w.daytime)
    LEFT OUTER JOIN holiday h ON (TRUNC(r.trip_time) = h.holiday_date);
```

檢視範例資料 (選擇性)

下列查詢會顯示資料表中的項目。您可以執行此操作，以確保資料表已正確製作。

```
SELECT *
FROM trip_data
LIMIT 5;
```

下列為上一個操作的輸出範例。

```
+---------------------+-----------+----------+---------+-----------+----------
+-------------+----------------+--------------+--------+-----------------
+------------+--------------+
|      trip_time      | trip_count | trip_hour | trip_day | trip_month | trip_year
 | trip_quarter | trip_month_week | trip_week_day | temp_c | precip_amount_mm |
 is_holiday | serial_number |
+---------------------+-----------+----------+---------+-----------+----------
+-------------+----------------+--------------+--------+-----------------
+------------+--------------+
| 2017-03-21 22:00:00 |         47 |        22 |       21 |          3 |        17 |
         1 |               3 |             3 |      1 |               0 |          0 |
             1 |
| 2018-05-04 01:00:00 |         19 |         1 |        4 |          5 |        18 |
         2 |               1 |             6 |     12 |               0 |          0 |
             3 |
| 2018-01-11 10:00:00 |         93 |        10 |       11 |          1 |        18 |
         1 |               2 |             5 |      9 |               0 |          0 |
             5 |
| 2017-10-28 04:00:00 |         20 |         4 |       28 |         10 |        17 |
         4 |               4 |             7 |     11 |               0 |          1 |
             7 |
| 2017-12-31 21:00:00 |         11 |        21 |       31 |         12 |        17 |
         4 |               5 |             1 |    -15 |               0 |          1 |
             9 |
+---------------------+-----------+----------+---------+-----------+----------
+-------------+----------------+--------------+--------+-----------------
+------------+--------------+
```

顯示屬性之間的關聯 (選擇性)

確定相關性有助於測量屬性之間的關聯強度。關聯層級可協助您判斷哪些因素會影響您的目標輸出。在此教學課程中，目標輸出為 trip_count。

下列查詢會建立或取代 sp_correlation 程序。您可以使用名為 sp_correlation 的預存程序來顯示 Amazon Redshift 中資料表內屬性之間的關聯性。

```
CREATE OR REPLACE PROCEDURE sp_correlation(source_schema_name in varchar(255),
 source_table_name in varchar(255), target_column_name in varchar(255),
 output_temp_table_name inout varchar(255)) AS $$
DECLARE
  v_sql varchar(max);
```

```
  v_generated_sql varchar(max);
  v_source_schema_name varchar(255)=lower(source_schema_name);
  v_source_table_name varchar(255)=lower(source_table_name);
  v_target_column_name varchar(255)=lower(target_column_name);
BEGIN
  EXECUTE 'DROP TABLE IF EXISTS ' || output_temp_table_name;
  v_sql = '
SELECT
  ''CREATE temp table '|| output_temp_table_name||' AS SELECT ''|| outer_calculation||
  '' FROM (SELECT COUNT(1) number_of_items, SUM('||v_target_column_name||')
 sum_target, SUM(POW('||v_target_column_name||',2)) sum_square_target, POW(SUM('||
v_target_column_name||'),2) square_sum_target,''||
  inner_calculation||
  '' FROM (SELECT ''||
  column_name||
  '' FROM '||v_source_table_name||''))''
FROM
  (
  SELECT
    DISTINCT
    LISTAGG(outer_calculation,'','') OVER () outer_calculation
    ,LISTAGG(inner_calculation,'','') OVER () inner_calculation
    ,LISTAGG(column_name,'','') OVER () column_name
  FROM
    (
    SELECT
      CASE WHEN atttypid=16 THEN ''DECODE(''||column_name||'',true,1,0)'' ELSE
 column_name END column_name
      ,atttypid
      ,''CAST(DECODE(number_of_items * sum_square_''||rn||'' - square_sum_''||
rn||'',0,null,(number_of_items*sum_target_''||rn||'' - sum_target * sum_''||rn||
        '')/SQRT((number_of_items * sum_square_target - square_sum_target) *
 (number_of_items * sum_square_''||rn||
        '' - square_sum_''||rn||''))) AS numeric(5,2)) ''||column_name
 outer_calculation
      ,''sum(''||column_name||'') sum_''||rn||'',''||
            ''SUM(trip_count*''||column_name||'') sum_target_''||rn||'',''||
            ''SUM(POW(''||column_name||'',2)) sum_square_''||rn||'',''||
            ''POW(SUM(''||column_name||''),2) square_sum_''||rn inner_calculation
    FROM
      (
      SELECT
        row_number() OVER (order by a.attnum) rn
        ,a.attname::VARCHAR column_name
```

```
        ,a.atttypid
      FROM pg_namespace AS n
        INNER JOIN pg_class AS c ON n.oid = c.relnamespace
        INNER JOIN pg_attribute AS a ON c.oid = a.attrelid
      WHERE a.attnum > 0
        AND n.nspname = '''||v_source_schema_name||'''
        AND c.relname = '''||v_source_table_name||'''
        AND a.atttypid IN (16,20,21,23,700,701,1700)
      )
    )
)';
  EXECUTE v_sql INTO v_generated_sql;
  EXECUTE  v_generated_sql;
END;
$$ LANGUAGE plpgsql;
```

下面的查詢顯示目標列、trip_count 和我們的資料集中的其他數字屬性之間的相關性。

```
call sp_correlation(
    'public',
    'trip_data',
    'trip_count',
    'tmp_corr_table'
);

SELECT
    *
FROM
    tmp_corr_table;
```

下列範例顯示上一個 sp_correlation 操作的輸出。

```
+------------+-----------+----------+------------+-----------+--------------
+-----------------+---------------+--------+------------------+------------
+---------------+
| trip_count | trip_hour | trip_day | trip_month | trip_year | trip_quarter
 | trip_month_week | trip_week_day | temp_c | precip_amount_mm | is_holiday |
 serial_number |
+------------+-----------+----------+------------+-----------+--------------
+-----------------+---------------+--------+------------------+------------
+---------------+
|          1 |      0.32 |     0.01 |       0.18 |      0.12 |         0.18 |
     0 |          0.02 |   0.53 |            -0.07 |      -0.13 |             0 |
```

```
+------------+-----------+----------+------------+-----------+--------------
+-----------------+---------------+--------+------------------+------------
+---------------+
```

步驟 2：建立機器學習模型

1. 下列查詢會將您的資料分割為訓練集和驗證集，並指定 80% 的資料集用於訓練，而 20% 的資料集用於驗證。訓練集是 ML 模型的輸入，用於識別模型的最佳演算法。建立模型之後，您可以使用驗證集來驗證模型的準確度。

```
CREATE TABLE training_data AS
SELECT
    trip_count,
    trip_hour,
    trip_day,
    trip_month,
    trip_year,
    trip_quarter,
    trip_month_week,
    trip_week_day,
    temp_c,
    precip_amount_mm,
    is_holiday
FROM
    trip_data
WHERE
    serial_number > (
        SELECT
            COUNT(1) * 0.2
        FROM
            trip_data
    );

CREATE TABLE validation_data AS
SELECT
    trip_count,
    trip_hour,
    trip_day,
    trip_month,
    trip_year,
    trip_quarter,
    trip_month_week,
    trip_week_day,
```

```
    temp_c,
    precip_amount_mm,
    is_holiday,
    trip_time
FROM
    trip_data
WHERE
    serial_number <= (
        SELECT
            COUNT(1) * 0.2
        FROM
            trip_data
    );
```

2. 下列查詢會建立迴歸模型，以預測任何輸入日期和時間的 trip_count 值。在下列範例中，請將 *DOC-EXAMPLE-BUCKET* 取代為您的 S3 儲存貯體。

```
CREATE MODEL predict_rental_count
FROM
    training_data TARGET trip_count FUNCTION predict_rental_count
    IAM_ROLE default
    PROBLEM_TYPE regression
    OBJECTIVE 'mse'
    SETTINGS (
        s3_bucket '<DOC-EXAMPLE-BUCKET>',
        s3_garbage_collect off,
        max_runtime 5000
    );
```

步驟 3：驗證模型

1. 使用下列查詢來輸出模型的各個層面，並在輸出中尋找均方誤差指標。均方誤差是迴歸問題的典型準確指標。

```
show model predict_rental_count;
```

2. 針對驗證資料執行下列預測查詢，以比較預測的行程計數與實際的行程計數。

```
SELECT
    trip_time,
    actual_count,
```

```
    predicted_count,
    (actual_count - predicted_count) difference
FROM
    (
        SELECT
            trip_time,
            trip_count AS actual_count,
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
                trip_year,
                trip_quarter,
                trip_month_week,
                trip_week_day,
                temp_c,
                precip_amount_mm,
                is_holiday
            ) predicted_count
        FROM
            validation_data
    )
LIMIT
    5;
```

3. 下列查詢會根據驗證資料計算均方誤差和均方根誤差。您可以使用均方誤差和均方根誤差來測量預測數值目標與實際數值答案之間的距離。一個好的模型在這兩個指標中的分數都很低。下列查詢會傳回這兩個指標的值。

```
SELECT
    ROUND(
        AVG(POWER((actual_count - predicted_count), 2)),
        2
    ) mse,
    ROUND(
        SQRT(AVG(POWER((actual_count - predicted_count), 2))),
        2
    ) rmse
FROM
    (
        SELECT
            trip_time,
            trip_count AS actual_count,
```

```
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
                trip_year,
                trip_quarter,
                trip_month_week,
                trip_week_day,
                temp_c,
                precip_amount_mm,
                is_holiday
            ) predicted_count
        FROM
            validation_data
    );
```

4. 下列查詢會針對 2017-01-01 的每個行程時間計算行程計數中的百分比誤差。查詢會將行程時間從百分比誤差最低的時間排序到百分比誤差最高的時間。

```
SELECT
    trip_time,
    CAST(ABS(((actual_count - predicted_count) / actual_count)) * 100 AS DECIMAL
 (7,2)) AS pct_error
FROM
    (
        SELECT
            trip_time,
            trip_count AS actual_count,
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
                trip_year,
                trip_quarter,
                trip_month_week,
                trip_week_day,
                temp_c,
                precip_amount_mm,
                is_holiday
            ) predicted_count
        FROM
            validation_data
    )
```

```
WHERE
   trip_time LIKE '2017-01-01 %%:%%:%%'
ORDER BY
   2 ASC;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon Redshift ML 的成本
- CREATE MODEL 操作
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

教學課程：使用線性學習程式建置迴歸模型

在本教學課程中，您會使用來自 Amazon S3 的資料建立線性學習程式模型，並使用 Amazon Redshift ML 透過該模型執行預測查詢。 SageMaker 線性學習器演算法可解決迴歸或多類別分類問題。若要進一步了解回歸和多類別分類問題，請參閱 Amazon SageMaker 開發人員指南中的機器學習範例的問題類型。在本教學課程中，您需解決迴歸問題。線性學習程式演算法可同時訓練許多模型，並自動判斷最佳化程度最高的模型。您可以在 Amazon Redshift 中使用創建模型操作，該操作會使用創建線性學習器模型， SageMaker 並將預測函數發送到 Amazon Redshift。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的線性學習演算法。

您可以使用 CREATE MODEL 命令來匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATE MODEL 操作，將訓練資料指定為資料表或 SELECT 陳述式。

線性學習程式模型可最佳化連續目標或離散目標。連續目標用於迴歸，而離散變數用於分類。某些方法會僅針對連續目標提供解決方案，例如迴歸方法。與單純的超參數最佳化技術 (例如 Naive Bayes 技術) 相比，線性學習程式演算法的速度更快。單純的最佳化技術會假定每個輸入變數都是獨立的。若要使用線性學習程式演算法，您必須提供代表輸入維度的資料欄，以及表示觀測值的資料列。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的線性學習演算法。

在本教學課程中，您將建置一個線性學習程式模型，以預測鮑魚的年齡。您可以在鮑魚資料集上使用 CREATE MODEL 命令來確定鮑魚的物理測量之間的關係。然後，您可以使用該模型來判斷鮑魚的年齡。

使用案例範例

您可以使用線性學習程式和 Amazon Redshift ML 來解決其他迴歸問題，例如預測房屋的價格。您還可以使用 Redshift ML 來預測將使用城市自行車租賃服務的人數。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：驗證模型

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的管理設定。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 Amazon Redshift 查詢編輯器 v2 來執行下列查詢。這些查詢會將範例資料載入 Redshift，並將資料分割為訓練集和驗證集。

1. 以下查詢會建立 abalone_dataset 資料表。

```
CREATE TABLE abalone_dataset (
    id INT IDENTITY(1, 1),
    Sex CHAR(1),
    Length float,
    Diameter float,
    Height float,
    Whole float,
    Shucked float,
    Viscera float,
    Shell float,
    Rings integer
);
```

2. 下列查詢會將 Amazon S3 中鮑魚資料集的範例資料複製到您先前在 Amazon Redshift 中建立的 abalone_dataset 資料表。

```
COPY abalone_dataset
FROM
    's3://redshift-ml-multiclass/abalone.csv' REGION 'us-east-1' IAM_ROLE default CSV
 IGNOREHEADER 1 NULL AS 'NULL';
```

3. 透過手動分割資料，您將能夠透過配置額外預測集來驗證模型的準確性。下面的查詢會將資料分成兩組。abalone_training 資料表用於訓練，abalone_validation 資料表用於驗證。

```
CREATE TABLE abalone_training as
SELECT
    *
FROM
    abalone_dataset
WHERE
    mod(id, 10) < 8;

CREATE TABLE abalone_validation as
SELECT
    *
FROM
    abalone_dataset
WHERE
    mod(id, 10) >= 8;
```

步驟 2：建立機器學習模型

在此步驟中，您可以使用 CREATE MODEL 陳述式來建立具有線性學習程式演算法的機器學習模型。

下列查詢會使用 S3 儲存貯體建立具有 CREATE MODEL 操作的線性學習程式模型。將 *DOC-EXAMPLE-BUCKET* 取代為您自己的 S3 儲存貯體。

```
CREATE MODEL model_abalone_ring_prediction
FROM
    (
        SELECT
            Sex,
            Length,
            Diameter,
```

```
            Height,
            Whole,
            Shucked,
            Viscera,
            Shell,
            Rings AS target_label
        FROM
            abalone_training
    ) TARGET target_label FUNCTION f_abalone_ring_prediction IAM_ROLE default
 MODEL_TYPE LINEAR_LEARNER PROBLEM_TYPE REGRESSION OBJECTIVE 'MSE' SETTINGS (
        S3_BUCKET 'DOC-EXAMPLE-BUCKET',
        MAX_RUNTIME 15000
    );
```

顯示模型訓練的狀態 (選擇性)

您可以使用 SHOW MODEL 命令來知道模型何時準備就緒。

使用下列查詢來監控模型訓練進度。

```
SHOW MODEL model_abalone_ring_prediction;
```

當模型準備就緒時，先前操作的輸出看起來應與下列範例類似。請注意，輸出會提供 validation:mse 指標，即均方誤差。在下一個步驟中，您將使用均方誤差來驗證模型的準確度。

```
+--------------------------
+-----------------------------------------------------------------------------------------------------
+
|        Model Name        |
 model_abalone_ring_prediction                                    |
+--------------------------
+-----------------------------------------------------------------------------------------------------
+
| Schema Name              | public
                                        |
| Owner                    | awsuser
                                        |
| Creation Time            | Thu, 30.06.2022 18:00:10
                                        |
| Model State              | READY
                                        |
| validation:mse           |
                                4.168633 |
```

```
| Estimated Cost           |
                                4.291608 |
|                          |
                                         |
| TRAINING DATA:           |
                                         |
| Query                    | SELECT SEX , LENGTH , DIAMETER , HEIGHT , WHOLE ,
 SHUCKED , VISCERA , SHELL, RINGS AS TARGET_LABEL |
|                          | FROM ABALONE_TRAINING
                                         |
| Target Column            | TARGET_LABEL
                                         |
|                          |
                                         |
| PARAMETERS:              |
                                         |
| Model Type               | linear_learner
                                         |
| Problem Type             | Regression
                                         |
| Objective                | MSE
                                         |
| AutoML Job Name          | redshiftml-20220630180010947843
                                         |
| Function Name            | f_abalone_ring_prediction
                                         |
| Function Parameters      | sex length diameter height whole shucked viscera shell
                                         |
| Function Parameter Types | bpchar float8 float8 float8 float8 float8 float8 float8
                                         |
| IAM Role                 | default-aws-iam-role
                                         |
| S3 Bucket                | DOC-EXAMPLE-BUCKET
                                         |
| Max Runtime              |
                                15000 |
+--------------------------
+-----------------------------------------------------------------------------------------------------------
+
```

步驟 3：驗證模型

1. 下列預測查詢會計算均方誤差和均方根誤差，藉此驗證 abalone_validation 資料集上模型的準確性。

```
SELECT
    ROUND(AVG(POWER((tgt_label - predicted), 2)), 2) mse,
    ROUND(SQRT(AVG(POWER((tgt_label - predicted), 2))), 2) rmse
FROM
    (
        SELECT
            Sex,
            Length,
            Diameter,
            Height,
            Whole,
            Shucked,
            Viscera,
            Shell,
            Rings AS tgt_label,
            f_abalone_ring_prediction(
                Sex,
                Length,
                Diameter,
                Height,
                Whole,
                Shucked,
                Viscera,
                Shell
            ) AS predicted,
            CASE
                WHEN tgt_label = predicted then 1
                ELSE 0
            END AS match,
            CASE
                WHEN tgt_label <> predicted then 1
                ELSE 0
            END AS nonmatch
        FROM
            abalone_validation
    ) t1;
```

先前查詢的輸出看起來應該像下列範例。均方誤差指標的值應與 SHOW MODEL 操作輸出所顯示的 validation:mse 指標相似。

```
+-----+--------------------+
| mse |        rmse        |
+-----+--------------------+
| 5.1 | 2.2600000000000002 |
+-----+--------------------+
```

2. 使用下列查詢，在預測函數上執行 EXPLAIIN_MODEL 操作。操作會傳回模型可解釋性報告。如需 EXPLAIN_MODEL 操作的相關資訊，請參閱《Amazon Redshift 資料庫開發人員指南》中的 EXPLAIN_MODEL 函數。

```
SELECT
    EXPLAIN_MODEL ('model_abalone_ring_prediction');
```

下列資訊是先前 EXPLAIN_MODEL 操作所產生模型可解釋性報告的範例。每個輸入的值為 Shapley 值。Shapley 值代表每個輸入對模型預測的影響，值較高的輸入對預測的影響較大。在此範例中，值較高的輸入對預測鮑魚的年齡有較大的影響。

```
{
    "explanations": {
        "kernel_shap": {
            "label0": {
                "expected_value" :10.290688514709473,
                "global_shap_values": {
                    "diameter" :0.6856910187882492,
                    "height" :0.4415323937124035,
                    "length" :0.21507476107609084,
                    "sex" :0.448611774505744,
                    "shell" :1.70426496893776,
                    "shucked" :2.1181392924386994,
                    "viscera" :0.342220754059912,
                    "whole" :0.6711906974084011
                }
            }
        }
    },
    "version" :"1.0"
};
```

3. 針對尚未成熟的鮑魚，使用以下查詢來計算模型做出正確預測的百分比。未成熟的鮑魚有 10 個或更少的環，而正確預測的準確度與實際環數僅差不到一個環。

```
SELECT
    TRUNC(
        SUM(
            CASE
                WHEN ROUND(
                    f_abalone_ring_prediction(
                        Sex,
                        Length,
                        Diameter,
                        Height,
                        Whole,
                        Shucked,
                        Viscera,
                        Shell
                    ),
                    0
                ) BETWEEN Rings - 1
                AND Rings + 1 THEN 1
                ELSE 0
            END
        ) / CAST(COUNT(SHELL) AS FLOAT),
        4
    ) AS prediction_pct
FROM
    abalone_validation
WHERE
    Rings <= 10;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon Redshift ML 的成本
- CREATE MODEL 操作
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

教學課程：使用線性學習程式建置多類別分類模型

在本教學課程中，您會使用來自 Amazon S3 的資料建立線性學習程式模型，然後使用 Amazon Redshift ML 透過該模型執行預測查詢。 SageMaker 線性學習器演算法可解決迴歸或分類問題。若要進一步了解回歸和多類別分類問題，請參閱 Amazon SageMaker 開發人員指南中的機器學習範例的問題類型。在本教學課程中，您將解決多類別分類問題。線性學習程式演算法可同時訓練許多模型，並自動判斷最佳化程度最高的模型。您可以在 Amazon Redshift 中使用創建模型操作，該操作會使用創建線性學習器模型， SageMaker 並將預測函數發送到 Amazon Redshift。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的線性學習演算法。

您可以使用 CREATE MODEL 命令來匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATE MODEL 操作，將訓練資料指定為資料表或 SELECT 陳述式。

線性學習程式模型可最佳化連續目標或離散目標。連續目標用於迴歸，而離散變數用於分類。某些方法會僅針對連續目標提供解決方案，例如迴歸方法。與單純的超參數最佳化技術 (例如 Naive Bayes 技術) 相比，線性學習程式演算法的速度更快。單純的最佳化技術會假定每個輸入變數都是獨立的。線性學習程式演算法可同時訓練許多模型，並選取最佳化程度最高的模型。一個類似的算法是 XGBoost，它結合了一組更簡單和模型更弱的估計來做出預測。若要進一步了解 XGBoost，請參閱 Amazon 開發人員指南中的 XGBoost 演算法。 SageMaker

若要使用線性學習程式演算法，您必須提供代表輸入維度的資料欄，以及表示觀測值的資料列。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的線性學習演算法。

在本教學課程中，您將建置一個線性學習程式模型，以預測指定區域的覆蓋類型。您可以在 UCI 機器學習儲存庫的 Covertype 資料集上使用 CREATE MODEL 命令。然後，您可以使用命令建立的預測函數來判斷荒野區域中的覆蓋類型。森林覆蓋類型通常是一個樹種。Redshift ML 將用於建立模型的輸入包括土壤類型、與道路的距離以及荒野區域指定。如需資料集的相關資訊，請參閱 UCI 機器學習儲存庫中的 Covertype 資料集。

使用案例範例

您可以使用 Amazon Redshift ML 解決線性學習程式的其他多類別分類問題，例如從影像預測植物的種類。您還可以預測客戶將購買的產品數量。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：驗證模型

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的管理設定。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 Amazon Redshift 查詢編輯器 v2 來執行下列查詢。這些查詢會將範例資料載入 Redshift，並將資料分割為訓練集和驗證集。

1. 以下查詢會建立 `covertype_data` 資料表。

```
CREATE TABLE public.covertype_data (
    elevation bigint ENCODE az64,
    aspect bigint ENCODE az64,
    slope bigint ENCODE az64,
    horizontal_distance_to_hydrology bigint ENCODE az64,
    vertical_distance_to_hydrology bigint ENCODE az64,
    horizontal_distance_to_roadways bigint ENCODE az64,
    hillshade_9am bigint ENCODE az64,
    hillshade_noon bigint ENCODE az64,
    hillshade_3pm bigint ENCODE az64,
    horizontal_distance_to_fire_points bigint ENCODE az64,
    wilderness_area1 bigint ENCODE az64,
    wilderness_area2 bigint ENCODE az64,
    wilderness_area3 bigint ENCODE az64,
    wilderness_area4 bigint ENCODE az64,
    soil_type1 bigint ENCODE az64,
    soil_type2 bigint ENCODE az64,
    soil_type3 bigint ENCODE az64,
    soil_type4 bigint ENCODE az64,
    soil_type5 bigint ENCODE az64,
    soil_type6 bigint ENCODE az64,
    soil_type7 bigint ENCODE az64,
    soil_type8 bigint ENCODE az64,
    soil_type9 bigint ENCODE az64,
    soil_type10 bigint ENCODE az64,
    soil_type11 bigint ENCODE az64,
```

```
    soil_type12 bigint ENCODE az64,
    soil_type13 bigint ENCODE az64,
    soil_type14 bigint ENCODE az64,
    soil_type15 bigint ENCODE az64,
    soil_type16 bigint ENCODE az64,
    soil_type17 bigint ENCODE az64,
    soil_type18 bigint ENCODE az64,
    soil_type19 bigint ENCODE az64,
    soil_type20 bigint ENCODE az64,
    soil_type21 bigint ENCODE az64,
    soil_type22 bigint ENCODE az64,
    soil_type23 bigint ENCODE az64,
    soil_type24 bigint ENCODE az64,
    soil_type25 bigint ENCODE az64,
    soil_type26 bigint ENCODE az64,
    soil_type27 bigint ENCODE az64,
    soil_type28 bigint ENCODE az64,
    soil_type29 bigint ENCODE az64,
    soil_type30 bigint ENCODE az64,
    soil_type31 bigint ENCODE az64,
    soil_type32 bigint ENCODE az64,
    soil_type33 bigint ENCODE az64,
    soil_type34 bigint ENCODE az64,
    soil_type35 bigint ENCODE az64,
    soil_type36 bigint ENCODE az64,
    soil_type37 bigint ENCODE az64,
    soil_type38 bigint ENCODE az64,
    soil_type39 bigint ENCODE az64,
    soil_type40 bigint ENCODE az64,
    cover_type bigint ENCODE az64
) DISTSTYLE AUTO;
```

2. 下列查詢會將 Amazon S3 中 Covertype 資料集的範例資料複製到您先前在 Amazon Redshift 中建立的 covertype_data 資料表。

```
COPY public.covertype_data
FROM
    's3://redshift-ml-multiclass/covtype.data.gz' IAM_ROLE DEFAULT gzip DELIMITER ','
 REGION 'us-east-1';
```

3. 透過手動分割資料，您將能夠透過配置額外測試集來驗證模型的準確性。下面的查詢會將資料分成三組。covertype_training 資料表用於訓練、covertype_validation 資料表用於驗證，而 covertype_test 資料表用於測試您的模型。您將使用訓練集來訓練模型，並使用驗證集來驗證模

型的開發。然後，您可以使用測試集來測試模型的效能，並查看模型與資料集之間是否過度擬合或低度擬合。

```
CREATE TABLE public.covertype_data_prep AS
SELECT
    a.*,
    CAST (random() * 100 AS int) AS data_group_id
FROM
    public.covertype_data a;

--training dataset
 CREATE TABLE public.covertype_training as
SELECT
    *
FROM
    public.covertype_data_prep
WHERE
    data_group_id < 80;

--validation dataset
 CREATE TABLE public.covertype_validation AS
SELECT
    *
FROM
    public.covertype_data_prep
WHERE
    data_group_id BETWEEN 80
    AND 89;

--test dataset
 CREATE TABLE public.covertype_test AS
SELECT
    *
FROM
    public.covertype_data_prep
WHERE
    data_group_id > 89;
```

步驟 2：建立機器學習模型

在此步驟中，您可以使用 CREATE MODEL 陳述式來建立具有線性學習程式演算法的機器學習模型。

下列查詢會使用 S3 儲存貯體建立具有 CREATE MODEL 操作的線性學習程式模型。將 *DOC-EXAMPLE-BUCKET* 取代為您自己的 S3 儲存貯體。

```
CREATE MODEL forest_cover_type_model
FROM
    (
        SELECT
            Elevation,
            Aspect,
            Slope,
            Horizontal_distance_to_hydrology,
            Vertical_distance_to_hydrology,
            Horizontal_distance_to_roadways,
            HIllshade_9am,
            Hillshade_noon,
            Hillshade_3pm,
            Horizontal_Distance_To_Fire_Points,
            Wilderness_Area1,
            Wilderness_Area2,
            Wilderness_Area3,
            Wilderness_Area4,
            soil_type1,
            Soil_Type2,
            Soil_Type3,
            Soil_Type4,
            Soil_Type5,
            Soil_Type6,
            Soil_Type7,
            Soil_Type8,
            Soil_Type9,
            Soil_Type10,
            Soil_Type11,
            Soil_Type12,
            Soil_Type13,
            Soil_Type14,
            Soil_Type15,
            Soil_Type16,
            Soil_Type17,
            Soil_Type18,
            Soil_Type19,
            Soil_Type20,
            Soil_Type21,
            Soil_Type22,
```

```
            Soil_Type23,
            Soil_Type24,
            Soil_Type25,
            Soil_Type26,
            Soil_Type27,
            Soil_Type28,
            Soil_Type29,
            Soil_Type30,
            Soil_Type31,
            Soil_Type32,
            Soil_Type33,
            Soil_Type34,
            Soil_Type36,
            Soil_Type37,
            Soil_Type38,
            Soil_Type39,
            Soil_Type40,
            Cover_type
        from
            public.covertype_training
    ) TARGET cover_type FUNCTION predict_cover_type IAM_ROLE default MODEL_TYPE
 LINEAR_LEARNER PROBLEM_TYPE MULTICLASS_CLASSIFICATION OBJECTIVE 'Accuracy' SETTINGS (
        S3_BUCKET '<DOC-EXAMPLE-BUCKET>',
        S3_GARBAGE_COLLECT OFF,
        MAX_RUNTIME 15000
    );
```

顯示模型訓練的狀態 (選擇性)

您可以使用 SHOW MODEL 命令來知道模型何時準備就緒。

使用下列查詢來監控模型訓練進度。

```
SHOW MODEL forest_cover_type_model;
```

當模型準備就緒時，先前操作的輸出看起來應與下列範例類似。請注意，輸出會提供 validation:multiclass_accuracy 指標，您可以在下列範例的右側檢視該指標。多類別準確度可測量模型正確分類的資料點百分比。在下一個步驟中，您將使用多類別準確度來驗證模型的準確度。

```
+---------------------------------
+-----------------------------------------------------------------------------------------------------------------
+
```

```
|              Key               |

 Value

                                                          |
+--------------------------------
+-------------------------------------------------------------------------------------------------------
+
| Model Name                     | forest_cover_type_model

                                                          |
| Schema Name                    | public

                                                          |
| Owner                          | awsuser

                                                          |
```

```
| Creation Time                   | Tue, 12.07.2022 20:24:32

                                                          |
| Model State                     | READY

                                                          |
| validation:multiclass_accuracy |

                                                  0.724952 |
| Estimated Cost                  |

                                                  5.341750 |
|                                 |
```

```
                                                    |
| TRAINING DATA:                 |

                                                    |
| Query                          | SELECT ELEVATION, ASPECT, SLOPE,
 HORIZONTAL_DISTANCE_TO_HYDROLOGY, VERTICAL_DISTANCE_TO_HYDROLOGY,
 HORIZONTAL_DISTANCE_TO_ROADWAYS, HILLSHADE_9AM, HILLSHADE_NOON, HILLSHADE_3PM ,
 HORIZONTAL_DISTANCE_TO_FIRE_POINTS, WILDERNESS_AREA1, WILDERNESS_AREA2,
 WILDERNESS_AREA3, WILDERNESS_AREA4, SOIL_TYPE1, SOIL_TYPE2, SOIL_TYPE3, SOIL_TYPE4,
 SOIL_TYPE5, SOIL_TYPE6, SOIL_TYPE7, SOIL_TYPE8, SOIL_TYPE9, SOIL_TYPE10 , SOIL_TYPE11,
 SOIL_TYPE12 , SOIL_TYPE13 , SOIL_TYPE14, SOIL_TYPE15, SOIL_TYPE16, SOIL_TYPE17,
 SOIL_TYPE18, SOIL_TYPE19, SOIL_TYPE20, SOIL_TYPE21, SOIL_TYPE22, SOIL_TYPE23,
 SOIL_TYPE24, SOIL_TYPE25, SOIL_TYPE26, SOIL_TYPE27, SOIL_TYPE28, SOIL_TYPE29,
 SOIL_TYPE30, SOIL_TYPE31, SOIL_TYPE32, SOIL_TYPE33, SOIL_TYPE34, SOIL_TYPE36,
 SOIL_TYPE37, SOIL_TYPE38, SOIL_TYPE39, SOIL_TYPE40, COVER_TYPE |
|                                | FROM PUBLIC.COVERTYPE_TRAINING

                                                    |
| Target Column                  | COVER_TYPE
```

```
                                                    |
|                                 |

                                                    |
| PARAMETERS:                     |

                                                    |
| Model Type                      | linear_learner

                                                    |
| Problem Type                    | MulticlassClassification

                                                    |
```

```
| Objective                       | Accuracy

                                                  |
| AutoML Job Name                 | redshiftml-20220712202432187659

                                                  |
| Function Name                   | predict_cover_type

                                                  |
| Function Parameters             | elevation aspect slope
 horizontal_distance_to_hydrology vertical_distance_to_hydrology
 horizontal_distance_to_roadways hillshade_9am hillshade_noon hillshade_3pm
 horizontal_distance_to_fire_points wilderness_area1 wilderness_area2 wilderness_area3
 wilderness_area4 soil_type1 soil_type2 soil_type3 soil_type4 soil_type5 soil_type6
 soil_type7 soil_type8 soil_type9 soil_type10 soil_type11 soil_type12 soil_type13
 soil_type14 soil_type15 soil_type16 soil_type17 soil_type18 soil_type19 soil_type20
 soil_type21 soil_type22 soil_type23 soil_type24 soil_type25 soil_type26 soil_type27
 soil_type28 soil_type29 soil_type30 soil_type31 soil_type32 soil_type33 soil_type34
 soil_type36 soil_type37 soil_type38 soil_type39 soil_type40
                                                |
| Function Parameter Types        | int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
```

```
 int8 int8 int8 int8 int8 int8 int8 int8 int8
                                                                                |
| IAM Role                        | default-aws-iam-role
                                                                                |
| S3 Bucket                       | DOC-EXAMPLE-BUCKET
                                                                                |
| Max Runtime                     |
                                                                          15000 |
+---------------------------------
+-------------------------------------------------------------------------------------------------------
+
```

步驟 3：驗證模型

1. 下列預測查詢會透過計算多類別準確度來驗證 covertype_validation 資料集上模型的準確度。多類別準確度是模型預測正確的百分比。

```
SELECT
    CAST(sum(t1.match) AS decimal(7, 2)) AS predicted_matches,
    CAST(sum(t1.nonmatch) AS decimal(7, 2)) AS predicted_non_matches,
    CAST(sum(t1.match + t1.nonmatch) AS decimal(7, 2)) AS total_predictions,
    predicted_matches / total_predictions AS pct_accuracy
FROM
    (
        SELECT
            Elevation,
            Aspect,
            Slope,
            Horizontal_distance_to_hydrology,
            Vertical_distance_to_hydrology,
            Horizontal_distance_to_roadways,
            HIllshade_9am,
            Hillshade_noon,
            Hillshade_3pm,
            Horizontal_Distance_To_Fire_Points,
            Wilderness_Area1,
            Wilderness_Area2,
            Wilderness_Area3,
            Wilderness_Area4,
            soil_type1,
            Soil_Type2,
            Soil_Type3,
            Soil_Type4,
            Soil_Type5,
            Soil_Type6,
            Soil_Type7,
            Soil_Type8,
            Soil_Type9,
            Soil_Type10,
            Soil_Type11,
            Soil_Type12,
            Soil_Type13,
            Soil_Type14,
            Soil_Type15,
            Soil_Type16,
```

```
            Soil_Type17,
            Soil_Type18,
            Soil_Type19,
            Soil_Type20,
            Soil_Type21,
            Soil_Type22,
            Soil_Type23,
            Soil_Type24,
            Soil_Type25,
            Soil_Type26,
            Soil_Type27,
            Soil_Type28,
            Soil_Type29,
            Soil_Type30,
            Soil_Type31,
            Soil_Type32,
            Soil_Type33,
            Soil_Type34,
            Soil_Type36,
            Soil_Type37,
            Soil_Type38,
            Soil_Type39,
            Soil_Type40,
            Cover_type AS actual_cover_type,
            predict_cover_type(
                Elevation,
                Aspect,
                Slope,
                Horizontal_distance_to_hydrology,
                Vertical_distance_to_hydrology,
                Horizontal_distance_to_roadways,
                HIllshade_9am,
                Hillshade_noon,
                Hillshade_3pm,
                Horizontal_Distance_To_Fire_Points,
                Wilderness_Area1,
                Wilderness_Area2,
                Wilderness_Area3,
                Wilderness_Area4,
                soil_type1,
                Soil_Type2,
                Soil_Type3,
                Soil_Type4,
                Soil_Type5,
```

```
            Soil_Type6,
            Soil_Type7,
            Soil_Type8,
            Soil_Type9,
            Soil_Type10,
            Soil_Type11,
            Soil_Type12,
            Soil_Type13,
            Soil_Type14,
            Soil_Type15,
            Soil_Type16,
            Soil_Type17,
            Soil_Type18,
            Soil_Type19,
            Soil_Type20,
            Soil_Type21,
            Soil_Type22,
            Soil_Type23,
            Soil_Type24,
            Soil_Type25,
            Soil_Type26,
            Soil_Type27,
            Soil_Type28,
            Soil_Type29,
            Soil_Type30,
            Soil_Type31,
            Soil_Type32,
            Soil_Type33,
            Soil_Type34,
            Soil_Type36,
            Soil_Type37,
            Soil_Type38,
            Soil_Type39,
            Soil_Type40
        ) AS predicted_cover_type,
        CASE
            WHEN actual_cover_type = predicted_cover_type THEN 1
            ELSE 0
        END AS match,
        CASE
            WHEN actual_cover_type <> predicted_cover_type THEN 1
            ELSE 0
        END AS nonmatch
    FROM
```

```
            public.covertype_validation
    ) t1;
```

先前查詢的輸出看起來應該像下列範例。多類別準確度指標的值應與 SHOW MODEL 操作輸出所顯示的 validation:multiclass_accuracy 指標相似。

```
+-------------------+-----------------------+-------------------+--------------+
| predicted_matches | predicted_non_matches | total_predictions | pct_accuracy |
+-------------------+-----------------------+-------------------+--------------+
|             41211 |                 16324 |             57535 |   0.71627704 |
+-------------------+-----------------------+-------------------+--------------+
```

2. 下面的查詢會預測 wilderness_area2 中最常見的覆蓋類型。此資料集包括四個荒野區域和七種覆蓋類型。一個荒野區域可以有多種覆蓋類型。

```
SELECT t1. predicted_cover_type, COUNT(*)
FROM
(
SELECT
   Elevation,
   Aspect,
   Slope,
   Horizontal_distance_to_hydrology,
   Vertical_distance_to_hydrology,
   Horizontal_distance_to_roadways,
   HIllshade_9am,
   Hillshade_noon,
   Hillshade_3pm ,
   Horizontal_Distance_To_Fire_Points,
   Wilderness_Area1,
   Wilderness_Area2,
   Wilderness_Area3,
   Wilderness_Area4,
   soil_type1,
   Soil_Type2,
   Soil_Type3,
   Soil_Type4,
   Soil_Type5,
   Soil_Type6,
   Soil_Type7,
   Soil_Type8,
   Soil_Type9,
```

```
    Soil_Type10 ,
    Soil_Type11,
    Soil_Type12 ,
    Soil_Type13 ,
    Soil_Type14,
    Soil_Type15,
    Soil_Type16,
    Soil_Type17,
    Soil_Type18,
    Soil_Type19,
    Soil_Type20,
    Soil_Type21,
    Soil_Type22,
    Soil_Type23,
    Soil_Type24,
    Soil_Type25,
    Soil_Type26,
    Soil_Type27,
    Soil_Type28,
    Soil_Type29,
    Soil_Type30,
    Soil_Type31,
    Soil_Type32,
    Soil_Type33,
    Soil_Type34,
    Soil_Type36,
    Soil_Type37,
    Soil_Type38,
    Soil_Type39,
    Soil_Type40,
    predict_cover_type( Elevation,
    Aspect,
    Slope,
    Horizontal_distance_to_hydrology,
    Vertical_distance_to_hydrology,
    Horizontal_distance_to_roadways,
    HIllshade_9am,
    Hillshade_noon,
    Hillshade_3pm ,
    Horizontal_Distance_To_Fire_Points,
    Wilderness_Area1,
    Wilderness_Area2,
    Wilderness_Area3,
    Wilderness_Area4,
```

```
   soil_type1,
   Soil_Type2,
   Soil_Type3,
   Soil_Type4,
   Soil_Type5,
   Soil_Type6,
   Soil_Type7,
   Soil_Type8,
   Soil_Type9,
   Soil_Type10,
   Soil_Type11,
   Soil_Type12,
   Soil_Type13,
   Soil_Type14,
   Soil_Type15,
   Soil_Type16,
   Soil_Type17,
   Soil_Type18,
   Soil_Type19,
   Soil_Type20,
   Soil_Type21,
   Soil_Type22,
   Soil_Type23,
   Soil_Type24,
   Soil_Type25,
   Soil_Type26,
   Soil_Type27,
   Soil_Type28,
   Soil_Type29,
   Soil_Type30,
   Soil_Type31,
   Soil_Type32,
   Soil_Type33,
   Soil_Type34,
   Soil_Type36,
   Soil_Type37,
   Soil_Type38,
   Soil_Type39,
   Soil_Type40) AS predicted_cover_type

FROM public.covertype_test
WHERE wilderness_area2 = 1)
t1
```

```
GROUP BY 1;
```

先前操作的輸出看起來應與下列範例類似。該輸出意味著該模型預測大部分覆蓋是覆蓋類型 1，並且有一些覆蓋類型 2 和 7。

```
+----------------------+-------+
| predicted_cover_type | count |
+----------------------+-------+
|                    2 |   564 |
|                    7 |    97 |
|                    1 |  2309 |
+----------------------+-------+
```

3. 下列查詢會顯示單一荒野區域中最常見的覆蓋類型。查詢會顯示該覆蓋類型的數量和覆蓋類型的荒野區域。

```
SELECT t1. predicted_cover_type, COUNT(*), wilderness_area
FROM
(
SELECT
   Elevation,
   Aspect,
   Slope,
   Horizontal_distance_to_hydrology,
   Vertical_distance_to_hydrology,
   Horizontal_distance_to_roadways,
   HIllshade_9am,
   Hillshade_noon,
   Hillshade_3pm ,
   Horizontal_Distance_To_Fire_Points,
   Wilderness_Area1,
   Wilderness_Area2,
   Wilderness_Area3,
   Wilderness_Area4,
   soil_type1,
   Soil_Type2,
   Soil_Type3,
   Soil_Type4,
   Soil_Type5,
   Soil_Type6,
   Soil_Type7,
   Soil_Type8,
```

```
    Soil_Type9,
    Soil_Type10 ,
    Soil_Type11,
    Soil_Type12 ,
    Soil_Type13 ,
    Soil_Type14,
    Soil_Type15,
    Soil_Type16,
    Soil_Type17,
    Soil_Type18,
    Soil_Type19,
    Soil_Type20,
    Soil_Type21,
    Soil_Type22,
    Soil_Type23,
    Soil_Type24,
    Soil_Type25,
    Soil_Type26,
    Soil_Type27,
    Soil_Type28,
    Soil_Type29,
    Soil_Type30,
    Soil_Type31,
    Soil_Type32,
    Soil_Type33,
    Soil_Type34,
    Soil_Type36,
    Soil_Type37,
    Soil_Type38,
    Soil_Type39,
    Soil_Type40,
    predict_cover_type( Elevation,
    Aspect,
    Slope,
    Horizontal_distance_to_hydrology,
    Vertical_distance_to_hydrology,
    Horizontal_distance_to_roadways,
    HIllshade_9am,
    Hillshade_noon,
    Hillshade_3pm ,
    Horizontal_Distance_To_Fire_Points,
    Wilderness_Area1,
    Wilderness_Area2,
    Wilderness_Area3,
```

```
    Wilderness_Area4,
    soil_type1,
    Soil_Type2,
    Soil_Type3,
    Soil_Type4,
    Soil_Type5,
    Soil_Type6,
    Soil_Type7,
    Soil_Type8,
    Soil_Type9,
    Soil_Type10,
    Soil_Type11,
    Soil_Type12,
    Soil_Type13,
    Soil_Type14,
    Soil_Type15,
    Soil_Type16,
    Soil_Type17,
    Soil_Type18,
    Soil_Type19,
    Soil_Type20,
    Soil_Type21,
    Soil_Type22,
    Soil_Type23,
    Soil_Type24,
    Soil_Type25,
    Soil_Type26,
    Soil_Type27,
    Soil_Type28,
    Soil_Type29,
    Soil_Type30,
    Soil_Type31,
    Soil_Type32,
    Soil_Type33,
    Soil_Type34,
    Soil_Type36,
    Soil_Type37,
    Soil_Type38,
    Soil_Type39,
    Soil_Type40) AS predicted_cover_type,
    CASE WHEN Wilderness_Area1 = 1 THEN 1
         WHEN Wilderness_Area2 = 1 THEN 2
         WHEN Wilderness_Area3 = 1 THEN 3
         WHEN Wilderness_Area4 = 1 THEN 4
```

```
        ELSE 0
   END AS wilderness_area

FROM public.covertype_test)
t1
GROUP BY 1, 3
ORDER BY 2 DESC
LIMIT 1;
```

先前操作的輸出看起來應與下列範例類似。

```
+----------------------+-------+-----------------+
| predicted_cover_type | count | wilderness_area |
+----------------------+-------+-----------------+
|                    2 | 15738 |               1 |
+----------------------+-------+-----------------+
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- 使用 Amazon Redshift ML 的成本
- CREATE MODEL 操作
- EXPLAIN_MODEL 函數

如需機器學習的相關資訊，請參閱下列文件：

- 機器學習概述
- 適用於新手和專家的機器學習
- 什麼是機器學習預測的公平性和模型解釋性？

調校查詢效能

Amazon Redshift 使用基於結構化查詢語言 (SQL) 的查詢來與系統中的資料與物件互動。資料處理語言 (DML) 是 SQL 的子集，您它用來檢視、新增、變更和刪除資料。資料定義語言 (DDL) 是 SQL 的子集，您用它來新增、變更和刪除資料庫物件 (例如：資料表和檢視表)。

設定系統之後，通常您最常使用的會是 DML，尤其是使用 SELECT 命令來擷取和檢視資料。若要在 Amazon Redshift 中編寫有效的資料擷取查詢，請熟悉 SELECT 語法，並套用設計資料表的 Amazon Redshift 最佳實務中所述的秘訣，將查詢的效率盡量放大。

若要了解 Amazon Redshift 如何處理查詢，請使用查詢處理和分析和改善查詢小節。然後您可以套用此資訊結合診斷工具來識別和消除查詢效能中的問題。

若要識別和解決使用 Amazon Redshift 查詢時可能遇到的部分最常見和最嚴重的問題，請使用對查詢進行故障診斷小節。

主題

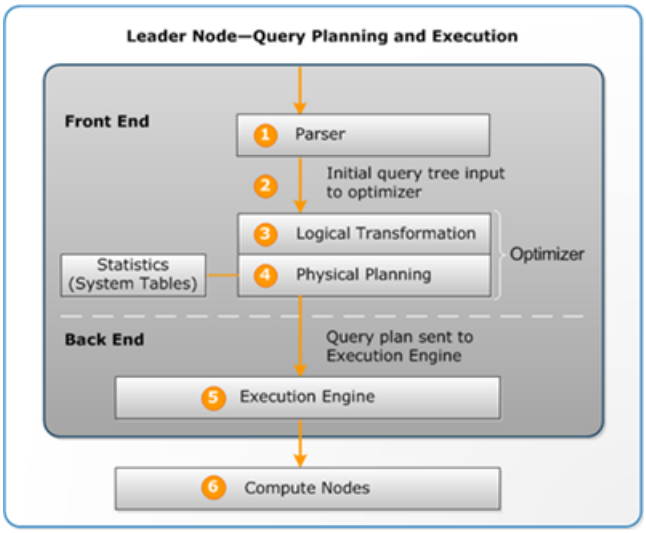
查詢處理

Amazon Redshift 會透過剖析器和最佳化器遞送提交的 SQL 查詢，以開發查詢計畫。執行引擎接著會將查詢計畫轉譯為程式碼，並將該程式碼傳送至運算節點以供執行。

主題

查詢計劃和執行工作流程

下表提供查詢計畫和執行工作流程的整體檢視。



查詢計畫和執行工作流程會遵循這些步驟：

1. 領導者節點接收查詢並剖析 SQL。
2. 剖析器會產生初始查詢樹狀目錄，它是原始查詢的邏輯呈現。然後，Amazon Redshift 會將此查詢樹狀目錄輸入至查詢最佳化器。
3. 最佳化器會進行評估，並且在必要時重新編寫查詢以最佳化其效率。此程序有時會造成建立多個相關的查詢來取代單一查詢。
4. 最佳化器會產生一個查詢計畫 (或數個，如果先前的步驟產生多個查詢) 以利用最佳效能執行。查詢計畫會指定執行選項，例如：聯結類型、聯結順序、彙整選項和資料配送需求。

   您可以使用 EXPLAIN 命令來檢視查詢計畫。查詢計畫為用於分析和調校複雜查詢的基礎工具。如需詳細資訊，請參閱 查詢計劃。
5. 執行引擎會將查詢計畫轉譯為步驟、區段和串流：

   步驟

   每個步驟為查詢執行期間所需的個別操作。您可以結合步驟，以允許運算節點執行查詢、聯結或其他資料庫操作。

   區段

   可以透過單一程序完成的數個步驟組合，也是可由運算節點配量執行的最小的編譯單位。配量是 Amazon Redshift 中平行處理的單位。串流中的區段會平行執行。

   串流

   要透過可用運算節點配量打包的區段的集合。

執行引擎會根據步驟、區段和串流產生編譯的程式碼。相較於轉譯的程式碼，編譯的程式碼執行得更快速，並且使用較少運算容量。然後將此編譯的程式碼播送至運算節點。

> **Note**
>
> 設定查詢的基準時，應該一律比較查詢的第二個執行的時間，因為第一個執行時間包含編譯程式碼的間接。如需詳細資訊，請參閱 影響查詢效能的因素。

6. 運算節點配量會平行執行查詢區段。隨著此程序，Amazon Redshift 會利用最佳化的網路通訊、記憶體和磁碟管理，在查詢計畫步驟之間傳遞中繼結果。這也有助於加快查詢執行速度。

步驟 5 和 6 會對每個串流發生一次。引擎會為一個串流建立可執行的區段，並將它們傳送至運算節點。當該串流的區段完成時，引擎會產生下一個串流的區段。以此方式，引擎可以分析前一個串流中發生的動作 (例如，操作是否基於磁碟) 以影響下一個串流中區段的產生。

當運算節點完成時，它們會將查詢結果傳回至領導者節點供最終處理。領導者節點會將資料合併至單一結果集，並解決任何需要的排序或彙整。然後領導者節點會將結果傳回至用戶端。

> **Note**
>
> 在查詢執行期間，運算節點可能會在必要時傳回一些資料至領導者節點。例如，如果您的子查詢具有 LIMIT 子句，則會於資料在叢集間重新配送供進一步處理之前，將限制套用到領導者節點。

查詢計劃

您可以使用查詢計畫來取得執行查詢所需的個別操作的相關資訊。使用查詢計畫之前，建議您先了解 Amazon Redshift 如何處理處理中的查詢和建立查詢計畫。如需詳細資訊，請參閱 查詢計劃和執行工作流程。

若要建立查詢計畫，請執行 EXPLAIN 命令，後面接著實際的查詢文字。查詢計畫可提供下列資訊：

- 執行引擎將執行的操作，請由下至上閱讀結果。
- 每個操作會執行的步驟類型。
- 每個操作中使用的資料表和資料欄。
- 每個操作中處理的資料量，就資料列的數量和資料寬度 (位元組) 而言。

- 操作的相對成本。成本是一種測量方式，會比較計畫內步驟的相對執行時間。成本不會提供實際執行時間或記憶體消耗的任何精確資訊，也不會提供執行計畫之間有意義的比較。但它可提供您查詢中耗費最多資源的操作的指示。

EXPLAIN 命令不會實際執行查詢。它只會顯示如果查詢是在目前操作條件下執行時，Amazon Redshift 會執行的計畫。如果您變更資料表的結構描述或資料，並再次執行 ANALYZE 以更新統計資訊中繼資料，查詢計畫可能不同。

EXPLAIN 的查詢計畫輸出為查詢執行的簡化、高階檢視。它不會說明平行查詢處理的詳細資訊。若要查看詳細資訊，請執行查詢本身，然後從 SVL_QUERY_SUMMARY 或 SVL_QUERY_REPORT 檢視取得查詢摘要資訊。如需使用這些檢視的相關資訊，請參閱分析查詢摘要。

下列範例顯示 EVENT 資料表上簡易 GROUP BY 查詢的 EXPLAIN 輸出：

```
explain select eventname, count(*) from event group by eventname;

                            QUERY PLAN
------------------------------------------------------------------------
XN HashAggregate  (cost=131.97..133.41 rows=576 width=17)
  ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=17)
```

EXPLAIN 會傳回每個操作的下列指標：

費用

用在計畫內比較操作的相對值。成本由兩個小數值組成，以兩個句點分隔，例如 `cost=131.97..133.41`。第一個值，在此情況下的 131.97，提供傳回此操作第一個資料列的相對成本。第二個值，在此情況下的 133.41，提供完成操作的相對成本。當您讀取計劃時，查詢計劃中的成本是累計的，因此此範例中的 HashAggregate 成本 (131.97.. 133.41) 會包含其下方的「序號掃描」成本 (0.00.. 87.98)。

資料列

要傳回的估計資料列數量。在此範例中，掃描預期傳回 8798 個資料列。 HashAggregate 運算子本身預期會傳回 576 列 (在結果集中捨棄重複的事件名稱之後)。

> **Note**
>
> 資料列估計是根據 ANALYZE 命令產生的可用統計資料。如果最近未執行 ANALYZE，估計將較不可靠。

寬度

估計的平均資料列寬度 (位元組)。在此範例中，平均資料列的寬度預期為 17 個位元組。

EXPLAIN 運算子

此小節簡要描述您最常在 EXPLAIN 輸出中看見的運算子。如需運算子的完整清單，請參閱 SQL 命令小節中的 EXPLAIN。

循序掃描運算子

循序掃描運算子 (Seq Scan) 指出資料表掃描。Seq Scan 會從開頭到結尾循序掃描資料表中的每個資料欄，並評估每個資料列的查詢限制條件 (在 WHERE 子句中)。

聯結運算子

Amazon Redshift 會根據要聯結資料表的實體設計、聯結所需的位置資料，以及查詢本身的特定需求來選取聯結運算子。

- 巢狀迴路

  最差的最佳聯結，巢狀迴路主要用於交叉聯結 (笛卡兒乘積) 和一些對等聯結中。

- 雜湊聯結和雜湊

  雜湊聯結和雜湊通常會較巢狀迴路聯結更快速，用於內部聯結和左右外部聯結。當聯結資料表中的聯結資料欄不是散發索引鍵也不是排序索引鍵時，會使用這些運算子。雜湊運算子會為聯結中的內部資料表建立雜湊表；雜湊聯結運算子會讀取外部資料表、雜湊聯結資料欄，並在內部雜湊表中尋找相符項目。

- 合併聯結

  合併聯結通常是最快速的聯結，用於內部聯結和外部聯結。合併聯結不會用於完整聯結。當聯結資料表中的聯結資料欄為散發索引鍵也是排序索引鍵，並且少於 20% 的聯結資料表未排序時，會使用此運算子。它會依序讀取兩個排序的資料表，並尋找相符的資料列。若要檢視未排序資料列的百分比，請查詢 SVV_TABLE_INFO 系統資料表。

- 空間聯結

  通常基於空間資料的鄰近程度快速聯結，用於 GEOMETRY 和 GEOGRAPHY 資料類型。

彙整運算子

查詢計畫會在牽涉到彙整函數和 GROUP BY 操作的查詢中使用下列運算子。

- Aggregate

  純量彙整函數的運算子，例如 AVG 和 SUM。
- HashAggregate

  未排序的分組彙整函數的運算子。
- GroupAggregate

  排序的分組彙整函數的運算子。

排序運算子

查詢計畫會在查詢必須排序或合併結果集時使用下列運算子。

- 排序

  評估 ORDER BY 子句和其他排序操作，例如 UNION 查詢和聯結、SELECT DISTINCT 查詢和視窗函數要求的排序。
- 合併

  根據衍生自平行操作中繼排序的結果，產生最終排序的結果。

UNION、INTERSECT 和 EXCEPT 運算子

查詢計畫會對牽涉到 UNION、INTERSECT 和 EXCEPT 設定操作的查詢使用下列運算子。

- Subquery

  用來執行 UNION 查詢。
- Hash Intersect Distinct

  用來執行 INTERSECT 查詢。
- SetOp 除外

  用來執行 EXCEPT (或 MINUS) 查詢。

其他運算子

下列運算子也會經常出現在例行查詢的 EXPLAIN 輸出中。

- 唯一

  消除 SELECT DISTINCT 查詢和 UNION 查詢的重複項目。
- 限制

  處理 LIMIT 子句。
- 視窗

  執行視窗函數。
- 結果

  執行未牽涉任何資料表存取的純量函數。
- Subplan

  用於特定子查詢。
- 網路

  將中繼結果傳送至領導者節點供進一步處理。
- Materialize

  儲存資料列以輸入至巢狀迴路聯結和一些合併聯結。

EXPLAIN 中的聯結

取決於查詢和基礎資料表的結構，查詢最佳化器會使用不同的聯結類型來擷取資料表資料。EXPLAIN 輸出會參考聯結類型、使用的資料表，以及資料表資料在叢集間配送的方式，以描述查詢的處理方式。

聯結類型範例

下列範例顯示查詢最佳化器可以使用的不同聯結類型。查詢計畫中使用的聯結類型取決於所牽涉資料表的實體設計。

範例：雜湊聯結兩個資料表

下列查詢會聯結 CATID 資料欄上的 EVENT 和 CATEGORY。CATID 為 CATEGORY (但不是 EVENT) 的配送和排序索引鍵。執行雜湊聯結，EVENT 為外部資料表而 CATEGORY 為內部資料表。

因為 CATEGORY 為較小的資料表，規劃器會在查詢處理期間透過使用 DS_BCAST_INNER 播送其一份副本至運算節點。此範例中的聯結成本可說明計畫的多數累積成本。

```
explain select * from category, event where category.catid=event.catid;

                              QUERY PLAN
----------------------------------------------------------------------
 XN Hash Join DS_BCAST_INNER  (cost=0.14..6600286.07 rows=8798 width=84)
   Hash Cond: ("outer".catid = "inner".catid)
   ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=35)
   ->  XN Hash  (cost=0.11..0.11 rows=11 width=49)
         ->  XN Seq Scan on category  (cost=0.00..0.11 rows=11 width=49)
```

> **Note**
>
> EXPLAIN 輸出中運算子的對應縮排有時會指出那些操作並非彼此相依，因此可以平行開始。在前述範例中，EVENT 資料表上的掃描雖然已與雜湊操作對應，EVENT 掃描仍必須等候雜湊操作已完全完成為止。

範例：合併聯結兩個資料表

下列查詢也使用 SELECT *，但它會聯結 LISTID 資料欄上的 SALES 和 LISTING，其中的 LISTID 已設為這兩個資料表的配送和排序索引鍵。已選擇合併聯結，並且聯結 (DS_DIST_NONE) 不需要重新配送資料。

```
explain select * from sales, listing where sales.listid = listing.listid;
QUERY PLAN
------------------------------------------------------------------------------
XN Merge Join DS_DIST_NONE  (cost=0.00..6285.93 rows=172456 width=97)
  Merge Cond: ("outer".listid = "inner".listid)
  ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=44)
  ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=53)
```

下列範例示範相同查詢內不同類型的聯結。在上一個範例中，SALES 和 LISTING 已合併聯結，但第三個資料表 EVENT 必須與合併聯結的結果雜湊聯結。重申，雜湊聯結會衍生播送成本。

```
explain select * from sales, listing, event
where sales.listid = listing.listid and sales.eventid = event.eventid;
                                  QUERY PLAN
------------------------------------------------------------------------------
```

```
XN Hash Join DS_BCAST_INNER  (cost=109.98..3871130276.17 rows=172456 width=132)
  Hash Cond: ("outer".eventid = "inner".eventid)
  ->  XN Merge Join DS_DIST_NONE  (cost=0.00..6285.93 rows=172456 width=97)
        Merge Cond: ("outer".listid = "inner".listid)
        ->  XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=44)
        ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=53)
  ->  XN Hash  (cost=87.98..87.98 rows=8798 width=35)
        ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=35)
```

範例：聯結、彙整和排序

下列查詢會執行 SALES 和 EVENT 資料表的雜湊聯結，接著彙整和排序操作以說明分組的 SUM 函數和 ORDER BY 子句。初始的排序運算子會在運算節點上平行執行。然後 Network 運算子會傳送結果至領導者節點，在其中，Merge 運算子會產生最終排序的結果。

```
explain select eventname, sum(pricepaid) from sales, event
where sales.eventid=event.eventid group by eventname
order by 2 desc;
                                          QUERY PLAN
------------------------------------------------------------------------------
 XN Merge  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
  Merge Key: sum(sales.pricepaid)
  ->  XN Network  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
        Send to leader
        ->  XN Sort  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
              Sort Key: sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=2815366577.07..2815366578.51 rows=576
 width=27)
                    ->  XN Hash Join DS_BCAST_INNER  (cost=109.98..2815365714.80
 rows=172456 width=27)
                          Hash Cond: ("outer".eventid = "inner".eventid)
                          ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456
 width=14)
                          ->  XN Hash  (cost=87.98..87.98 rows=8798 width=21)
                                ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798
 width=21)
```

資料重新分佈

聯結的 EXPLAIN 輸出也可為在叢集間移動資料的方式指定可輔助聯結的方法。資料可以透過播送或重新配送來移動。在播送中，來自聯結一端的資料值會從每個運算節點複製到每個其他運算節點，使得每個運算節點最終有完整的資料副本。在重新配送中，參與的資料值是從其目前的配量傳送至新配量 (可

能在不同的節點上)。資料一般會重新配送以符合參與聯結的其他資料表的散發索引鍵 (如果該散發索引鍵是其中一個聯結資料欄)。如果任何資料表都不具備其中一個聯結資料欄上的散發索引鍵，則會配送兩個資料表，或是將內部資料表播送至每個節點。

EXPLAIN 輸出也會參考內部和外部資料表。會先掃描內部資料表，並出現在較接近查詢計畫底端的位置。內部資料表為對其探測相符項目的資料表。它通常位於記憶體中，且通常是雜湊的來源資料表，如有可能，會是要聯結的兩個資料表中較小的那個。外部資料表為要對內部資料表比對的資料列的來源。通常是從磁碟讀取。查詢最佳化器會根據來自 ANALYZE 命令最近一次執行的資料庫統計資料來選擇內部和外部資料表。查詢的 FROM 子句中資料表的順序，並不會決定哪個資料表為內部以及哪個為外部。

請在查詢計畫中使用下列屬性來識別資料將移動的方式，以輔助查詢：

- DS_BCAST_INNER

  將整個內部資料表的副本播送至所有運算節點。
- DS_DIST_ALL_NONE

  不需要重新配送，因為已使用 DISTSTYLE ALL 將內部資料表配送至每個節點。
- DS_DIST_NONE

  不重新配送任何資料表。共置聯結可行，因為會聯結對應的配量，而不需在節點間移動資料。
- DS_DIST_INNER

  重新配送內部資料表。
- DS_DIST_OUTER

  重新配送外部資料表。
- DS_DIST_ALL_INNER

  因為外部資料表使用 DISTSTYLE ALL，會將整個內部資料表重新配送至單一配量。
- DS_DIST_BOTH

  重新配送這兩個資料表。

檢閱查詢計劃步驟

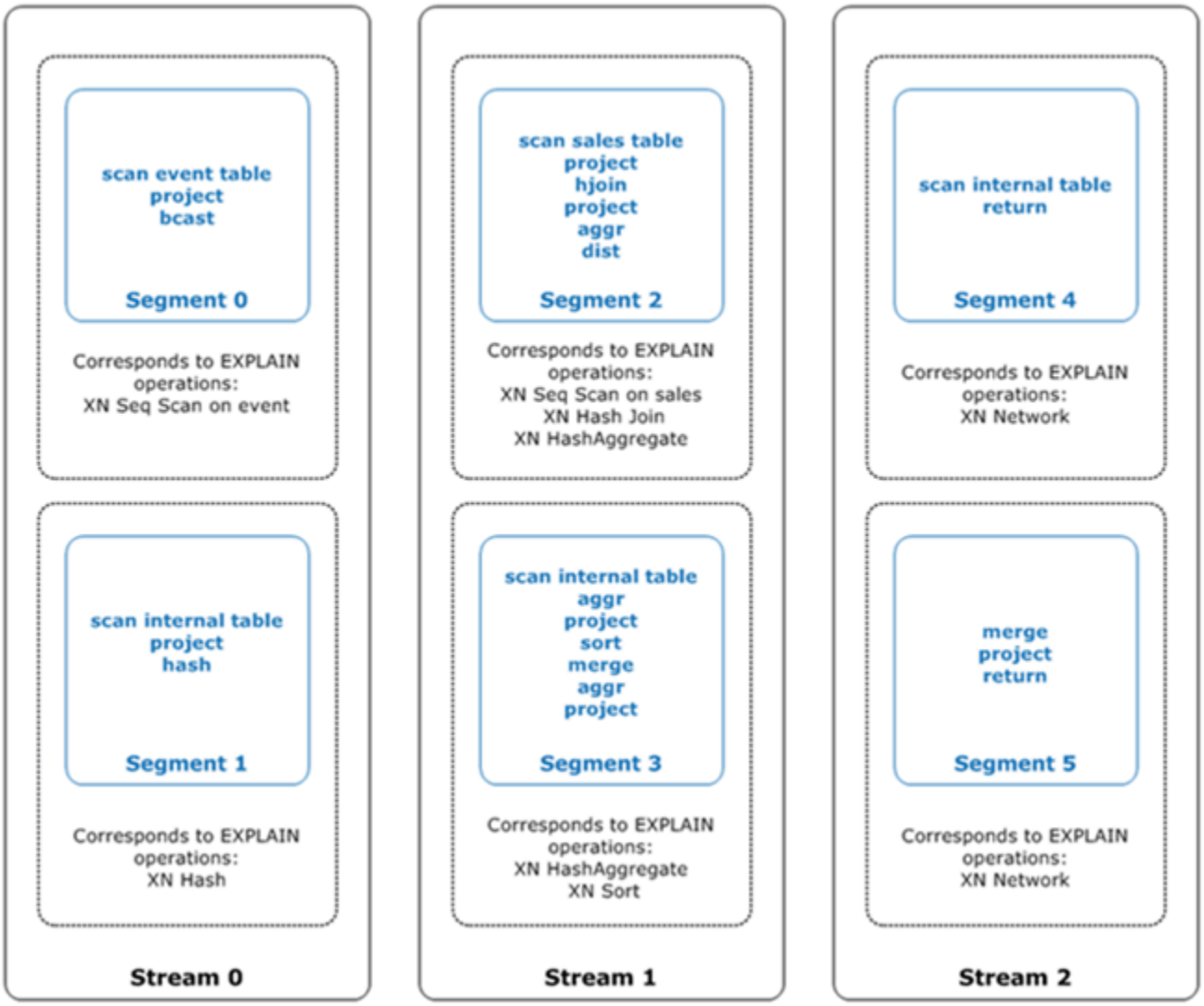
您可以執行 EXPLAIN 命令來查看查詢計畫中的步驟。下列範例顯示 SQL 查詢並說明輸出。您可以由下至上閱讀查詢計畫，以查看用於執行查詢的每個邏輯操作。如需詳細資訊，請參閱 查詢計劃。

```
explain
select eventname, sum(pricepaid) from sales, event
where sales.eventid = event.eventid
group by eventname
order by 2 desc;
```

```
XN Merge  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
  Merge Key: sum(sales.pricepaid)
  ->  XN Network  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
        Send to leader
        ->  XN Sort  (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
              Sort Key: sum(sales.pricepaid)
              ->  XN HashAggregate  (cost=2815366577.07..2815366578.51 rows=576
 width=27)
                    ->  XN Hash Join DS_BCAST_INNER  (cost=109.98..2815365714.80
 rows=172456 width=27)
                          Hash Cond: ("outer".eventid = "inner".eventid)
                          ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456
 width=14)
                          ->  XN Hash  (cost=87.98..87.98 rows=8798 width=21)
                                ->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798
 width=21)
```

在產生查詢計畫的過程中，查詢最佳化工具會將計畫劃分為串流、區段和步驟。查詢最佳化工具會細分計畫，以準備將資料和查詢工作負載分配到運算節點。如需串流、區段和步驟的相關資訊，請參閱查詢計劃和執行工作流程。

下圖顯示先前的查詢和相關聯的查詢計畫。其顯示涉及的查詢操作如何映射至 Amazon Redshift 用於為運算節點配量產生編譯程式碼的步驟。每個查詢計畫操作會映射至區段內的多個步驟，並且有時映射至串流內的多個區段。



在此圖中，查詢最佳化工具會執行查詢計畫，如下所示：

1. 在 Stream 0 中，查詢會以循序掃描操作執行 Segment 0，以掃描 events 資料表。查詢會使用雜湊操作繼續處理 Segment 1，為聯結中的內部資料表建立雜湊表。
2. 在 Stream 1 中，查詢會以循序掃描操作執行 Segment 2，以掃描 sales 資料表。它會使用雜湊連結繼續處理 Segment 2，以聯結含有不是散發索引鍵也不是排序索引鍵之聯結資料欄的資料表。它會再次使用雜湊彙總繼續處理 Segment 2，以彙總結果。然後查詢會執行 Segment 3 搭配雜湊彙總操作，以執行未排序的分組彙總函數和排序操作，進而評估 ORDER BY 子句和其他排序操作。
3. 在 Stream 2 中，查詢會在 Segment 4 和 Segment 5 中執行網路操作，以將中繼結果傳送到領導節點進一步處理。

查詢的最後一個區段會傳回資料。如果傳回集合已彙總或排序，則運算節點會各自將其中繼結果部分傳送到領導節點。領導節點會接著合併資料，以便將最終結果送回給請求的用戶端。

如需 EXPLAIN 運算子的相關資訊，請參閱 EXPLAIN。

影響查詢效能的因素

有一些因素會影響查詢效能。您的資料、叢集和資料庫操作的下列層面，都會影響處理查詢的速度。

- 節點、處理器或配量的數量 – 運算節點會分割為配量。更多節點表示更多處理器和更多配量，透過在配量間並行執行部分的查詢，可讓您的查詢處理得更快速。不過，更多節點也表示更大的開支，因此必須尋找適合您的系統的成本與效能的平衡點。如需 Amazon Redshift 叢集架構的相關資訊，請參閱資料倉儲系統架構。
- 節點類型 - Amazon Redshift 叢集可以使用數種節點類型之一。每個節點類型可提供不同的大小和限制，以幫助您適當地調整叢集的規模。節點大小會決定叢集中每個節點的儲存容量、記憶體、CPU 和價格。如需節點類型的相關資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 叢集概觀。
- 資料配送 – Amazon Redshift 會根據資料表的配送樣式，將資料表資料儲存在運算節點。當您執行查詢時，查詢最佳化器會視需要將資料重新配送至運算節點，以執行任何聯結與彙整。選擇資料表的適當配送樣式，有助於降低重新配送步驟所帶來的影響，方法是在執行聯結之前，將資料放置在需要的位置。如需詳細資訊，請參閱 使用資料分佈樣式。
- 資料排序排列 – Amazon Redshift 會根據資料表的排序索引鍵，以排序的順序將資料表資料儲存在磁碟上。查詢最佳化器和查詢處理器使用資料所在位置的資訊以減少必須掃描的區塊數量，藉以改善查詢速度。如需詳細資訊，請參閱 使用排序索引鍵。
- 資料集大小 – 由於需要掃描和重新配送更多資料列，因此叢集中較多的資料量可能拖慢查詢的查詢效能。您可以透過定期清空和封存資料，以及使用述詞來限制查詢資料集，藉以減緩此影響。
- 並行操作 – 一次執行多個操作可能影響查詢效能。每個操作會使用可用查詢佇列中的一或多個位置，並使用與那些位置關聯的記憶體。如果其他操作正在執行，則可能沒有足夠的查詢佇列位置可供使用。在此情況下，查詢必須等候位置開放之後才可以開始處理。如需建立和設定查詢佇列的相關資訊，請參閱實作工作負載管理。
- 查詢結構 – 寫入查詢的方式會影響其效能。請對程序寫入盡可能多的查詢，並傳回符合您需求的最少資料。如需詳細資訊，請參閱 設計查詢的 Amazon Redshift 最佳實務。
- 程式碼編譯 – Amazon Redshift 會為每個查詢執行計畫產生和編譯程式碼。

  編譯的程式碼執行得更快速，因為它省去了使用解譯器的間接成本。第一次產生和編譯程式碼時，通常會有些間接成本。因此，第一次執行查詢的效能可能會有偏差。執行一次性查詢時的間接成本可

能特別明顯。請進行第二次執行查詢，以便判斷其一般效能。Amazon Redshift 使用無伺服器編譯服務，將查詢編譯擴展到 Amazon Redshift 叢集的運算資源之外。編譯的程式碼區段會在叢集本機上快取，而且會在幾乎無限制的快取中快取。此快取會在叢集重新啟動後持續存在。相同查詢的後續執行可以加快執行速度，因為它可以略過編譯階段。

快取在 Amazon Redshift 版本之間不相容，因此會清除編譯快取，並在版本升級後執行查詢時重新編譯程式碼。如果您的查詢具有嚴格的 SLA，建議您預先執行查詢區段，以掃描叢集資料表中的資料。這可讓 Amazon Redshift 快取基礎資料表資料，減少版本升級後的查詢規劃時間。透過使用可擴展的編譯服務，Amazon Redshift 可以並行編譯程式碼以提供一致的快速效能。工作負載加速的程度取決於查詢的複雜性和並行性。

分析和改善查詢

從 Amazon Redshift 資料倉儲擷取資訊需要對非常大量的資料執行複雜的查詢，這可能耗費冗長時間來處理。若要確保盡可能快速處理查詢，有一些工具可供您用來識別潛在的效能問題。

主題

查詢分析工作流程

如果查詢耗費的時間超過預期，請使用下列步驟來識別並更正可能對查詢效能帶來負面影響的問題。如果您不確定系統中有哪些查詢可能透過效能調校而獲益，請在識別用於調校的最高候選項目查詢中執行診斷查詢以開始。

1. 確定您的資料表是根據最佳實務而設計。如需詳細資訊，請參閱 設計資料表的 Amazon Redshift 最佳實務。
2. 請查看是否可以刪除或封存資料表中任何不需要的資料。例如，假設您的查詢一律鎖定最近 6 個月的資料量，但是您的資料表中有最近 18 個月的資料。在此情況下，您可以刪除或封存較舊的資料，以減少需要掃描和配送的記錄數量。

3. 在查詢中對資料表執行 VACUUM 命令，以回收空間和重新排序資料列。如果未排序的區域很大，並且查詢使用聯結或述詞中的排序索引鍵，則執行 VACUUM 有幫助。
4. 在查詢中對資料表執行 ANALYZE 命令，以確定統計資料是最新的。如果查詢中任何資料表的大小最近有許多變更，則執行 ANALYZE 很有幫助。如果執行完整的 ANALYZE 命令將耗費太長時間，請對單一資料欄執行分析以減少處理時間。此方法仍會更新資料表大小統計資料；資料表大小是查詢計畫中的顯著因素。
5. 確定您的查詢已對每個類型的用戶端執行一次 (根據用戶端使用的連線通訊協定的類型而定)，使得該查詢會經過編譯和快取。這種方法會加速查詢的後續執行。如需詳細資訊，請參閱 影響查詢效能的因素。
6. 請檢查STL_ALERT_EVENT_LOG資料表來識別和更正您的查詢的可能問題。如需詳細資訊，請參閱 檢閱查詢提醒。
7. 執行 EXPLAIN 命令來取得查詢計畫，並用它來最佳化查詢。如需詳細資訊，請參閱 分析查詢計劃。
8. 使用SVL_QUERY_SUMMARY和SVL_QUERY_REPORT檢視來取得摘要資訊，並用它來最佳化查詢。如需詳細資訊，請參閱 分析查詢摘要。

有時應該快速執行的查詢會被強制等候其他長時間執行的查詢完成。在該情況下，對於查詢本身，您可能沒有可改善的項目，但您可以藉由對不同類型的查詢建立和使用查詢佇列來改善整體系統效能。若要獲得您的查詢佇列等候時間的概念，請參閱檢閱查詢的佇列等候時間。如需設定查詢佇列的相關資訊，請參閱實作工作負載管理。

檢閱查詢提醒

若要使用STL_ALERT_EVENT_LOG系統資料表來識別和更正您的查詢的潛在效能問題，請遵循這些步驟：

1. 執行下列動作來判斷您的查詢 ID：

```
select query, elapsed, substring
from svl_qlog
order by query
desc limit 5;
```

在 substring 欄位中檢查截斷的查詢文字，判斷要選取的 query 值。如果您已執行查詢超過一次，請使用來自具有較低 query 值資料列的 elapsed 值。那是編譯版本的資料列。如果您已執行許多查詢，您可以提升 LIMIT 子句使用的值，以確定您的查詢已包含在其中。

2. 從 STL_ALERT_EVENT_LOG 中，為您的查詢選取資料列：

```
Select * from stl_alert_event_log where query = MyQueryID;
```

userid	query	slice	segment	step	pid	xid	event	solution	event_time
100	32359	4	0	0	8780	71195	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-10 17:40:50
100	32359	5	0	0	8781	71195	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-10 17:40:50
100	109142	4	0	0	8780	302411	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-24 20:32:28
100	109142	5	0	0	8781	302411	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-24 20:32:28
100	109828	4	1	0	8746	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:27:52
100	109828	5	1	0	8747	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:27:52
100	109829	4	1	0	8760	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:28:01
100	109829	5	1	0	8761	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:28:01
100	113910	4	1	0	8774	316848	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-25 17:14:58

3. 評估您的查詢的結果。使用下表來找出任何遇到問題的潛在解決方案。

> **Note**
>
> STL_ALERT_EVENT_LOG 中不會有所有查詢的資料列，只有已發現問題的那些查詢。

問題	事件值	解決方案值	建議的解決方案
查詢中資料表的統計資料遺漏或過時。	遺漏查詢規劃器統計資料	執行 ANALYZE 命令	請參閱 資料表統計資訊遺漏或過時。
查詢計畫中有巢狀迴路聯結 (最差的最佳聯結)。	查詢計畫中的巢狀迴路聯結	檢閱聯結述詞以避免笛卡兒乘積	請參閱 巢狀迴圈。
掃描已略過相當多標示為已刪除但未清空的資料列，或已插入但未遞交的資料列。	已掃描大量刪除的資料列	執行 VACUUM 命令來回收刪除的空間	請參閱 幽靈資料列或未遞交的資料列。
已針對雜湊聯結或彙整重新配送超過 1,000,000 個資料列。	在網路上分散了大量的資 RowCount 料列：資料列已分散以處理彙總	檢閱散發索引鍵的選擇以共置聯結或彙整	請參閱 次佳資料分佈。
已針對雜湊聯結播送超過 1,000,000 個資料列。	在網路間播送大量資料列	檢閱散發索引鍵的選擇以共置聯	請參閱 次佳資料分佈。

問題	事件值	解決方案值	建議的解決方案
		結或並考慮使用配送資料表	
已在查詢計劃中指出 DS_DIST_ALL_INNER 重新配送樣式，其會強制序列執行，因為整個內部資料表已重新配送至單一節點。	查詢計畫中用於雜湊聯結的 DS_DIST_ALL_INNER	檢閱配送策略的選擇，以配送內部而非外部資料表	請參閱 次佳資料分佈。

分析查詢計劃

分析查詢計畫之前，您應該熟悉如何閱讀它。如果您對於閱讀查詢計畫不熟悉，建議您閱讀查詢計劃之後再繼續。

執行 EXPLAIN 命令來取得查詢計畫。若要分析查詢計畫提供的資料，請遵循這些步驟：

1. 識別具有最高成本的步驟。繼續進行其餘步驟時，請專注在那些。
2. 查看聯結類型：
   - 巢狀迴路：這類聯結的發生通常是因為省略了聯結條件。如需建議的解決方案，請參閱巢狀迴圈。
   - 雜湊和雜湊聯結：當聯結資料表中的聯結資料欄不是散發索引鍵也不是排序索引鍵時，會使用雜湊聯結。如需建議的解決方案，請參閱雜湊聯結。
   - 合併聯結：不需變更。
3. 注意哪個資料表用於內部聯結，以及哪個用於外部聯結。查詢引擎一般會對內部聯結選擇較小的資料表，以及對外部聯結選擇較大的資料表。如果這類選擇未發生，那麼，您的統計資料很可能過時。如需建議的解決方案，請參閱資料表統計資訊遺漏或過時。
4. 查看是否有任何高成本的排序操作。如果有，請參閱未排序或排序錯誤的資料列以取得建議的解決方案。
5. 尋找具有高成本操作的下列播送運算子：
   - DS_BCAST_INNER：表示資料表已廣播至所有運算節點。這對於小資料表來說很好，但對於較大的資料表來說並不理想。
   - DS_DIST_ALL_INNER：指出所有工作負載位於單一配量上。
   - DS_DIST_BOTH：指出繁重的重新配送。

   如需這些情況的建議解決方案，請參閱次佳資料分佈。

分析查詢摘要

若要取得較 EXPLAIN 所產生的查詢計畫更詳細的執行步驟和統計資料，請使用 SVL_QUERY_SUMMARY 和 SVL_QUERY_REPORT 系統檢視。

SVL_QUERY_SUMMARY 提供依串流的查詢統計資料。您可以使用其提供的資訊來識別具有代價高昂步驟、長時間執行步驟和寫入磁碟步驟的問題。

SVL_QUERY_REPORT 系統檢視可用來查看與 SVL_QUERY_SUMMARY 類似的資訊，但只能依運算節點配量而不能依串流查看。您可以使用配量層級資訊來偵測叢集間不平均的資料配送 (也稱為資料配送偏度)，它會強制部分節點執行較其他節點更多的工作，並影響查詢效能。

主題

- 使用 SVL_QUERY_SUMMARY 檢視
- 使用 SVL_QUERY_REPORT 檢視
- 將查詢計劃映射到查詢摘要

使用 SVL_QUERY_SUMMARY 檢視

若要依串流分析摘要資訊，請執行下列動作：

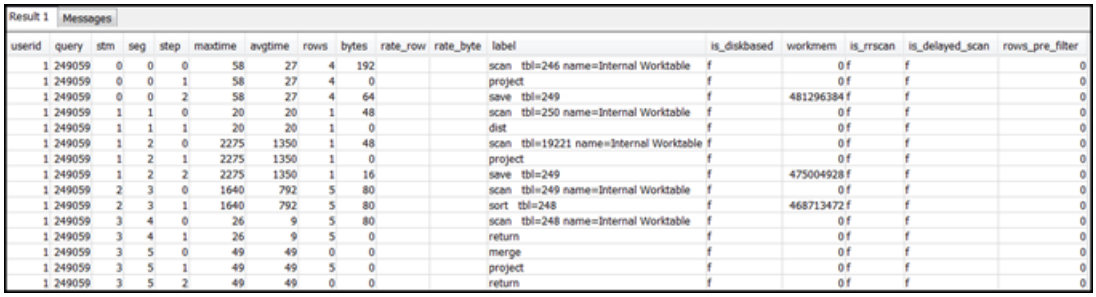
1. 執行下列查詢來判斷您的查詢 ID：

```
select query, elapsed, substring
from svl_qlog
order by query
desc limit 5;
```

   在 substring 欄位中檢查截斷的查詢文字，判斷代表您的查詢的 query 值。如果您已執行查詢超過一次，請使用來自具有較低 query 值資料列的 elapsed 值。那是編譯版本的資料列。如果您已執行許多查詢，您可以提升 LIMIT 子句使用的值，以確定您的查詢已包含在其中。

2. 從 SVL_QUERY_SUMMARY 中，為您的查詢選取資料列。依串流、區段和步驟排列結果：

```
select * from svl_query_summary where query = MyQueryID order by stm, seg, step;
```



Result 1 | Messages

userid	query	stm	seg	step	maxtime	avgtime	rows	bytes	rate_row	rate_byte	label	is_diskbased	workmem	is_rrscan	is_delayed_scan	rows_pre_filter
1	249059	0	0	0	58	27	4	192			scan tbl=246 name=Internal Worktable	f	0	f	f	0
1	249059	0	0	1	58	27	4	0			project	f	0	f	f	0
1	249059	0	0	2	58	27	4	64			save tbl=249	f	481296384	f	f	0
1	249059	1	1	0	20	20	1	48			scan tbl=250 name=Internal Worktable	f	0	f	f	0
1	249059	1	1	1	20	20	1	0			dist	f	0	f	f	0
1	249059	1	2	0	2275	1350	1	48			scan tbl=19221 name=Internal Worktable	f	0	f	f	0
1	249059	1	2	1	2275	1350	1	0			project	f	0	f	f	0
1	249059	1	2	2	2275	1350	1	16			save tbl=249	f	475004928	f	f	0
1	249059	2	3	0	1640	792	5	80			scan tbl=249 name=Internal Worktable	f	0	f	f	0
1	249059	2	3	1	1640	792	5	80			sort tbl=248	f	468713472	f	f	0
1	249059	3	4	0	26	9	5	80			scan tbl=248 name=Internal Worktable	f	0	f	f	0
1	249059	3	4	1	26	9	5	0			return	f	0	f	f	0
1	249059	3	5	0	49	49	0	0			merge	f	0	f	f	0
1	249059	3	5	1	49	49	5	0			project	f	0	f	f	0
1	249059	3	5	2	49	49	0	0			return	f	0	f	f	0

3. 使用將查詢計劃映射到查詢摘要中的資訊，將步驟與查詢計畫中的操作映射。它們的資料列和位元組應該具有大約相同的值 (查詢計畫中的資料列 * 寬度)。如果不同，請參閱資料表統計資訊遺漏或過時以取得建議的解決方案。
4. 查看任何步驟的 `is_diskbased` 欄位是否具有 `t` (true) 值。如果系統沒有為查詢處理配置足夠的記憶體，雜湊、彙整和排序為可能將資料寫入至磁碟的運算子。

   如果 `is_diskbased` 為 true，請參閱配置給查詢的記憶體不足以取得建議的解決方案。
5. 檢閱 `label` 欄位值，並查看步驟中的任何位置是否有 AGG-DIST-AGG 序列。出現它表示兩個步驟彙整，其代價高昂。若要修正此問題，請將 GROUP BY 子句變更為使用散發索引鍵 (如果有多個則第一個索引鍵)。
6. 檢閱每個區段的 `maxtime` 值 (區段中的所有步驟是相同的)。識別具有最高 `maxtime` 值的區段，並檢閱此區段中下列運算子的步驟。

> **Note**
>
> 高的 `maxtime` 值並不一定代表區段有問題。雖然值很高，但該區段可能並未耗費大量時間處理。串流中的所有區段會同時開始計時。不過，部分下游區段必須等到取得來自上游的資料後才能執行。此影響可能使得它們看起來耗費了長時間，因為其 `maxtime` 值將同時包含等候時間和處理時間。

- BCAST 或 DIST：在這些情況下，造成 `maxtime` 高值的原因可能是重新配送大量資料列。如需建議的解決方案，請參閱次佳資料分佈。
- HJOIN (雜湊聯結)：如果有問題步驟的 `rows` 欄位相較於查詢中最終 RETURN 步驟中的 `rows` 值有非常高的值，請參閱雜湊聯結以取得建議的解決方案。
- SCAN/SORT：在聯結步驟之前尋找步驟的 SCAN、SORT、SCAN、MERGE 序列。這個模式指出未排序的資料會經過掃描、排序，然後與資料表排序的區域合併。

查看相較於查詢中最終 RETURN 步驟的資料列值，SCAN 步驟的資料列值是否有非常高的值。這個模式指出執行引擎正在掃描稍後將捨棄的資料列，這樣做缺乏效率。如需建議的解決方案，請參閱不足的限制性述詞。

如果 SCAN 步驟的 maxtime 值很高，請參閱次佳的 WHERE 子句以取得建議的解決方案。

如果 SORT 步驟的 rows 值不是零，請參閱未排序或排序錯誤的資料列以取得建議的解決方案。

7. 檢閱最終 RETURN 步驟之前的 5–10 步驟的 rows 和 bytes 值，來了解傳回用戶端的資料量。此程序可以是一門藝術。

   例如，在下列查詢摘要中，您可以看到第三個 PROJECT 步驟提供 rows 值而非 bytes 值。查看前述步驟來找到具有相同 rows 值的步驟，您會找到同時提供資料列和位元組資訊的 SCAN 步驟：

userid	query	stm	seg	step	maxtime	avgtime	rows	bytes	rate_row	rate_byte	label	is_diskbased	workmem
1	187435	2	5	2	14307	12797	0	0			hash tbl=256	f	46871347
1	187435	3	6	0	531	308	387	229104			scan tbl=242 name=Internal Worktable	f	
1	187435	3	6	1	531	308	387	0			project	f	
1	187435	3	6	2	531	308	387	222912			save tbl=245	f	38063308
1	187435	4	7	0	390	390	0	0			scan tbl=238 name=Internal Worktable	f	
1	187435	4	7	1	390	390	0	0			dist	f	
1	187435	4	8	0	1218	1066	0	0			scan tbl=134954 name=Internal Worktable	f	
1	187435	4	8	1	1218	1066	0	0			project	f	
1	187435	4	8	2	1218	1066	0	0			save tbl=245	f	37434163
1	187435	5	9	0	171	83	387	222912			scan tbl=245 name=Internal Worktable	f	
1	187435	5	9	1	171	83	387	60120			dist	f	
1	187435	5	10	0	3579	3383	387	222912			scan tbl=134955 name=Internal Worktable	f	
1	187435	5	10	1	3579	3383	387	0			project	f	
1	187435	5	10	2	3579	3383	0	0			hjoin tbl=256	f	
1	187435	5	10	3	3579	3383	0	0			project	f	
1	187435	5	10	4	3579	3383	0	0			sort tbl=259	f	36805017
1	187435	6	11	0	10	7	0	0			scan tbl=259 name=Internal Worktable	f	
1	187435	6	11	1	10	7	0	0			return	f	
1	187435	6	12	0	9	9	0	0			merge	f	
1	187435	6	12	1	9	9	0	0			project	f	
1	187435	6	12	2	9	9	0	0			return	f	

   如果您要傳回異常大的資料量，請參閱非常大的結果集以取得建議的解決方案。

8. 相較於其他步驟，查看 bytes 值是否相對於任何步驟中的 rows 值來得高。這個模式可能指出您正選取許多資料欄。如需建議的解決方案，請參閱大型 SELECT 清單。

使用 SVL_QUERY_REPORT 檢視

若要依配量分析摘要資訊，請執行下列動作：

1. 執行下列動作來判斷您的查詢 ID：

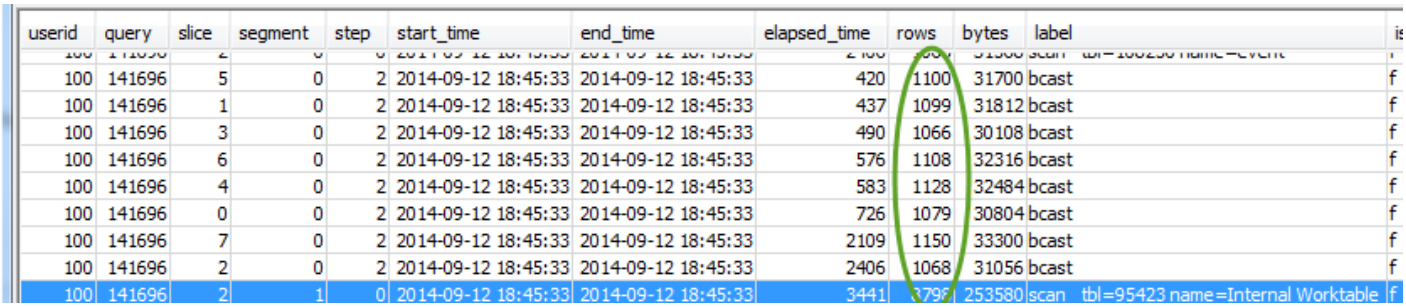
```
select query, elapsed, substring
from svl_qlog
order by query
desc limit 5;
```

在 substring 欄位中檢查截斷的查詢文字，判斷代表您的查詢的 query 值。如果您已執行查詢超過一次，請使用來自具有較低 query 值資料列的 elapsed 值。那是編譯版本的資料列。如果您已執行許多查詢，您可以提升 LIMIT 子句使用的值，以確定您的查詢已包含在其中。

2. 從 SVL_QUERY_REPORT 中，為您的查詢選取資料列。依區段、步驟、經過時間和資料列排列結果：

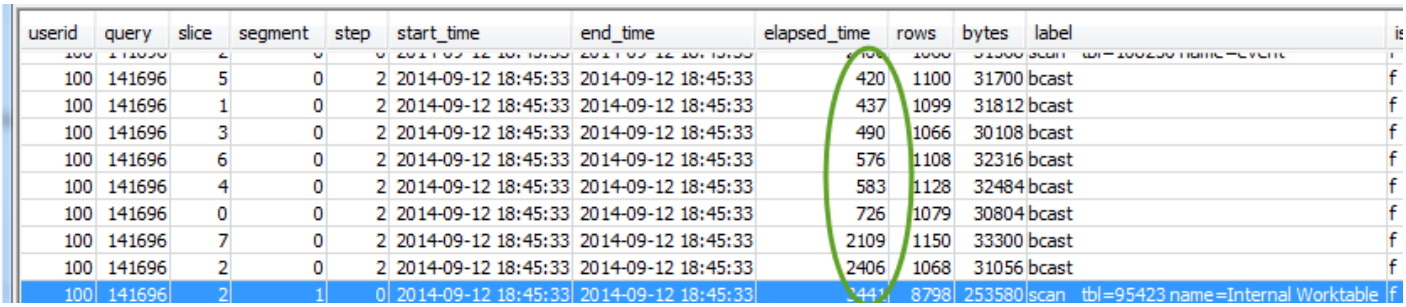
```
select * from svl_query_report where query = MyQueryID order by segment, step,
 elapsed_time, rows;
```

3. 針對每個步驟，查看所有配量是否正處理大約相同數量的資料列：



userid	query	slice	segment	step	start_time	end_time	elapsed_time	rows	bytes	label	is
100	141696	5	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	420	1100	31700	bcast	f
100	141696	1	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	437	1099	31812	bcast	f
100	141696	3	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	490	1066	30108	bcast	f
100	141696	6	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	576	1108	32316	bcast	f
100	141696	4	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	583	1128	32484	bcast	f
100	141696	0	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	726	1079	30804	bcast	f
100	141696	7	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	bcast	f
100	141696	2	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2406	1068	31056	bcast	f
100	141696	2	1	0	2014-09-12 18:45:33	2014-09-12 18:45:33	3441	3798	253580	scan tbl=95423 name=Internal Worktable	f

也查看所有配量是否正耗費大約相同的時間量：



userid	query	slice	segment	step	start_time	end_time	elapsed_time	rows	bytes	label	is
100	141696	5	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	420	1100	31700	bcast	f
100	141696	1	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	437	1099	31812	bcast	f
100	141696	3	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	490	1066	30108	bcast	f
100	141696	6	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	576	1108	32316	bcast	f
100	141696	4	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	583	1128	32484	bcast	f
100	141696	0	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	726	1079	30804	bcast	f
100	141696	7	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	bcast	f
100	141696	2	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2406	1068	31056	bcast	f
100	141696	2	1	0	2014-09-12 18:45:33	2014-09-12 18:45:33	3441	8798	253580	scan tbl=95423 name=Internal Worktable	f

這些值的差異大可能表示由於此特定查詢的次佳配送樣式造成的資料配送偏度。如需建議的解決方案，請參閱次佳資料分佈。

將查詢計劃映射到查詢摘要

它有助於將查詢計畫中的操作映射至查詢摘要中的步驟 (由標籤欄位值識別)，以取得更多詳細資訊：

查詢計劃操作	標籤欄位值	描述
Aggregate	AGGR	評估彙整函數和 GROUP BY 條件。

查詢計劃操作	標籤欄位值	描述
HashAggregate GroupAggregate		
DS_BCAST_INNER	BCAST (broadcast)	將整個資料表或某組資料列 (例如來自資料表篩選的一組資料列) 播送至所有節點。
不會出現在查詢計畫中	DELETE	刪除資料表中的資料列。
DS_DIST_NONE DS_DIST_ALL_NONE DS_DIST_INNER DS_DIST_ALL_INNER DS_DIST_ALL_BOTH	DIST (distribute)	針對平行聯結目的或其他平行處理，將資料列配送至節點。
HASH	HASH	建置要用於雜湊聯結的雜湊表。
雜湊聯結	HJOIN (hash join)	執行兩個資料表或中繼結果集的雜湊聯結。
不會出現在查詢計畫中	INSERT	將資料列插入至資料表。
限制	LIMIT	套用 LIMIT 子句至結果集。
合併	MERGE	合併衍生自平行排序或聯結操作的資料列。
合併聯結	MJOIN (merge join)	執行兩個資料表或中繼結果集的合併聯結。
巢狀迴路	NLOOP (nested loop)	執行兩個資料表或中繼結果集的巢狀迴路聯結。

查詢計劃操作	標籤欄位值	描述
不會出現在查詢計畫中	PARSE	將字串剖析為二進位值以進行載入。
專案	PROJECT	評估表達式。
網路	RETURN	將資料列傳回至領導者或用戶端。
不會出現在查詢計畫中	SAVE	具體化資料列以使於下一個處理步驟。
循序掃描	SCAN	掃描資料表或中繼結果集。
Sort	SORT	如其他後續操作 (例如聯結或彙整) 所要求排序資料列或中繼結果集，或滿足 ORDER BY 子句。
唯一	UNIQUE	如其他操作所要求套用 SELECT DISTINCT 子句或移除重複項目。
視窗	WINDOW	計算彙整和排名視窗函數。

改善 查詢效能

以下是影響查詢效能的一些常見問題，以及其診斷和解決方式的指示。

主題

- 資料表統計資訊遺漏或過時
- 巢狀迴圈
- 雜湊聯結
- 幽靈資料列或未遞交的資料列
- 未排序或排序錯誤的資料列
- 次佳資料分佈

- 配置給查詢的記憶體不足
- 次佳的 WHERE 子句
- 不足的限制性述詞
- 非常大的結果集
- 大型 SELECT 清單

資料表統計資訊遺漏或過時

如果資料表統計資料遺漏或過時，您可能會看見下列：

- EXPLAIN 命令結果中有警告訊息。
- STL_ALERT_EVENT_LOG 中遺漏統計資料提醒事件。如需詳細資訊，請參閱 檢閱查詢提醒。

若要修正此問題，請執行 ANALYZE。

巢狀迴圈

如果出現巢狀迴路，您可能會在 STL_ALERT_EVENT_LOG 中看見巢狀迴路提醒事件。您也可以透過執行識別具有巢狀迴圈的查詢中的查詢，來識別此類型的事件。如需詳細資訊，請參閱 檢閱查詢提醒。

若要修正此問題，請檢閱您的查詢以取得交叉聯結並在可能時加以移除。交叉聯結是沒有聯結條件的聯結，會造成兩個資料表的笛卡兒乘積。它們通常會以巢狀迴圈聯結的形式執行，這是可能的聯結類型中最慢的。

雜湊聯結

如果出現雜湊聯結，您可能會看見下列：

- 查詢計畫中的雜湊和雜湊聯結操作。如需詳細資訊，請參閱 分析查詢計劃。
- 區段中的 HJOIN 步驟具有 SVL_QUERY_SUMMARY 中最高的 maxtime 值。如需詳細資訊，請參閱 使用 SVL_QUERY_SUMMARY 檢視。

若要修正此問題，您可以採取兩個方法：

- 如果可能，重新寫入查詢以使用合併聯結。指定同時為散發索引鍵和排序索引鍵的聯結資料欄即可執行此動作。

- 如果 SVL_QUERY_SUMMARY 的 HJOIN 步驟的資料列欄位相較於查詢中最終 RETURN 步驟中的資料列值有非常高的值，請檢查您是否可以重新寫入查詢以在唯一資料欄上聯結。當查詢未在唯一資料欄上聯結時，例如主要索引鍵，它會增加聯結中牽涉的資料列數量。

幽靈資料列或未遞交的資料列

如果出現幽靈資料列或未遞交的資料列，您可能會在 STL_ALERT_EVENT_LOG 看見提醒事件，指出過量的幽靈資料列。如需詳細資訊，請參閱 檢閱查詢提醒。

若要修正此問題，您可以採取兩個方法：

- 查看 Amazon Redshift 主控台的載入索引標籤，以瞭解任何查詢資料表上的作用中載入操作。如果您看見作用中的載入操作，採取動作之前，請等候那些操作完成。
- 如果沒有作用中載入操作，請在查詢資料表上執行 VACUUM 來移除刪除的資料列。

未排序或排序錯誤的資料列

如果出現未排序或排序錯誤的資料列，您可能會在 STL_ALERT_EVENT_LOG 中看見非常高的篩選條件提醒事件。如需詳細資訊，請參閱 檢閱查詢提醒。

您也可以透過執行 識別具有資料扭曲或未排序資料列的資料表 中的查詢，查看您的查詢中是否任何資料表有大型未排序的區域。

若要修正此問題，您可以採取兩個方法：

- 在查詢資料表上執行 VACUUM 來重新排序資料列。
- 檢閱查詢資料表上的排序索引鍵，以查看是否有可以進行的任何改善。請記得評估此查詢的效能與其他重要查詢和整體系統的效能，再進行任何變更。如需詳細資訊，請參閱 使用排序索引鍵。

次佳資料分佈

如果資料配送為次佳，您可能會看見下列：

- STL_ALERT_EVENT_LOG 中出現序列執行、大型播送或大型配送提醒事件。如需詳細資訊，請參閱 檢閱查詢提醒。
- 針對指定步驟，配量未處理大約相同數量的資料列。如需詳細資訊，請參閱 使用 SVL_QUERY_REPORT 檢視。

- 針對指定步驟，配量未耗費大約相同的時間量。如需詳細資訊，請參閱 使用 SVL_QUERY_REPORT 檢視。

如果前述中無一成立，您也可以透過執行 識別具有資料扭曲或未排序資料列的資料表 中的查詢，查看您的查詢中是否有任何資料表有資料偏度。

若要修正此問題，查看查詢中資料表的分佈樣式，並確認是否可以進行任何改善。請記得評估此查詢的效能與其他重要查詢和整體系統的效能，再進行任何變更。如需詳細資訊，請參閱 使用資料分佈樣式。

配置給查詢的記憶體不足

如果對您的查詢配置了不足的記憶體，您可能會在 SVL_QUERY_SUMMARY 中看見某個步驟的 `is_diskbased` 值為 true。如需詳細資訊，請參閱 使用 SVL_QUERY_SUMMARY 檢視。

若要修正此問題，請透過暫時增加它所使用查詢位置的數量，為查詢配置更多記憶體。工作負載管理 (WLM) 會在查詢佇列中預留位置，大約等同於為佇列設定的並行層級。例如，具有並行層級 5 的佇列有 5 個位置。指派給佇列的記憶體會平均配置到每個位置。對一個查詢指派數個位置可讓該查詢存取所有那些位置的記憶體。如需如何暫時增加查詢位置的相關資訊，請參閱wlm_query_slot_count。

次佳的 WHERE 子句

如果您的 WHERE 子句造成過度的資料表掃描，您可能會在 SVL_QUERY_SUMMARY 中具有最高 `maxtime` 值的區段中看見 SCAN 步驟。如需詳細資訊，請參閱 使用 SVL_QUERY_SUMMARY 檢視。

若要修正此問題，請根據最大資料表的主要排序資料欄，將 WHERE 子句新增至查詢。這種方法可縮短降低掃描時間。如需詳細資訊，請參閱 設計資料表的 Amazon Redshift 最佳實務。

不足的限制性述詞

如果您的查詢有不足的限制性述詞，您可能會在 SVL_QUERY_SUMMARY 中具有最高 `maxtime` 值的區段中看見 SCAN 步驟，其相較於查詢中最終 RETURN 步驟中的 `rows` 值具有非常高的 `rows` 值。如需詳細資訊，請參閱 使用 SVL_QUERY_SUMMARY 檢視。

若要修正此問題，請嘗試新增述詞至查詢，或讓現有述詞更具限制性來縮小列輸出。

非常大的結果集

如果您的查詢傳回非常大的結果集，請考慮重新寫入查詢，以使用 UNLOAD 來寫入結果至 Amazon S3。此方式藉由利用平行處理，有效改善 RETURN 步驟的效能。如需檢查非常大型結果集的相關資訊，請參閱使用 SVL_QUERY_SUMMARY 檢視。

大型 SELECT 清單

如果您的查詢有異常大的 SELECT 清單，您可能會在 SVL_QUERY_SUMMARY 中看見相對於 (相較於其他步驟) 任何步驟的 bytes 值來得高的 rows 值。這個高 bytes 值可能是您正選取許多資料欄的指標。如需詳細資訊，請參閱 使用 SVL_QUERY_SUMMARY 檢視。

若要修正此問題，請檢閱您要選取的資料欄，並查看是否可移除任何項目。

診斷查詢以進行查詢調校

使用下列查詢來識別查詢的問題或可能影響查詢效能的基礎資料表。建議您使用這些查詢結合分析和改善查詢中討論的查詢調校程序。

主題

- 識別用於調校的最高候選項目查詢
- 識別具有資料扭曲或未排序資料列的資料表
- 識別具有巢狀迴圈的查詢
- 檢閱查詢的佇列等候時間
- 依資料表檢閱查詢提醒
- 識別具有遺漏統計資訊的資料表

識別用於調校的最高候選項目查詢

下列查詢會識別過去 7 天中已執行的前 50 個最耗時的陳述式。您可以使用結果來識別需要異常長時間的查詢。您也可以識別經常執行的查詢 (在結果集中出現超過一次的項目)。這些查詢往往是進行調校以改善系統效能的良好候選項目。

此查詢也提供與所識別的每個查詢關聯的提醒事件計數。這些提醒提供的詳細資訊可供您用來改善查詢的效能。如需詳細資訊，請參閱 檢閱查詢提醒。

```
select trim(database) as db, count(query) as n_qry,
max(substring (qrytext,1,80)) as qrytext,
```

```
min(run_minutes) as "min" ,
max(run_minutes) as "max",
avg(run_minutes) as "avg", sum(run_minutes) as total,
max(query) as max_query_id,
max(starttime)::date as last_run,
sum(alerts) as alerts, aborted
from (select userid, label, stl_query.query,
trim(database) as database,
trim(querytxt) as qrytext,
md5(trim(querytxt)) as qry_md5,
starttime, endtime,
(datediff(seconds, starttime,endtime)::numeric(12,2))/60 as run_minutes,
alrt.num_events as alerts, aborted
from stl_query
left outer join
(select query, 1 as num_events from stl_alert_event_log group by query ) as alrt
on alrt.query = stl_query.query
where userid <> 1 and starttime >=  dateadd(day, -7, current_date))
group by database, label, qry_md5, aborted
order by total desc limit 50;
```

識別具有資料扭曲或未排序資料列的資料表

下列查詢會識別具有不均資料配送 (資料偏度) 或高比例未排序資料列的資料表。

低的 skew 值指出資料表資料已正確配送。如果資料表有 4.00 或更高的 skew 值，請考慮修改其資料配送樣式。如需詳細資訊，請參閱 次佳資料分佈。

如果資料表的 pct_unsorted 值大於 20%，請考慮執行 VACUUM 命令。如需詳細資訊，請參閱 未排序或排序錯誤的資料列。

同時另外檢閱每個資料表的 mbytes 和 pct_of_total 值。這些資料欄會識別資料表的大小，以及資料表耗用的原始磁碟空間百分比。原始磁碟空間包括 Amazon Redshift 保留供內部使用的空間，因此會大於名目磁碟容量，它是可供使用者使用的磁碟空間容量。使用此資訊來確保您的可用磁碟空間至少是您最大資料表的 2.5 倍。有此可用空間可讓系統在處理複雜查詢時將中繼結果寫入至磁碟。

```
select trim(pgn.nspname) as schema,
trim(a.name) as table, id as tableid,
decode(pgc.reldiststyle,0, 'even',1,det.distkey ,8,'all') as distkey,
 dist_ratio.ratio::decimal(10,4) as skew,
det.head_sort as "sortkey",
det.n_sortkeys as "#sks", b.mbytes,
```

```
decode(b.mbytes,0,0,((b.mbytes/part.total::decimal)*100)::decimal(5,2)) as
 pct_of_total,
decode(det.max_enc,0,'n','y') as enc, a.rows,
decode( det.n_sortkeys, 0, null, a.unsorted_rows ) as unsorted_rows ,
decode( det.n_sortkeys, 0, null, decode( a.rows,0,0, (a.unsorted_rows::decimal(32)/
a.rows)*100) )::decimal(5,2) as pct_unsorted
from (select db_id, id, name, sum(rows) as rows,
sum(rows)-sum(sorted_rows) as unsorted_rows
from stv_tbl_perm a
group by db_id, id, name) as a
join pg_class as pgc on pgc.oid = a.id
join pg_namespace as pgn on pgn.oid = pgc.relnamespace
left outer join (select tbl, count(*) as mbytes
from stv_blocklist group by tbl) b on a.id=b.tbl
inner join (select attrelid,
min(case attisdistkey when 't' then attname else null end) as "distkey",
min(case attsortkeyord when 1 then attname  else null end ) as head_sort ,
max(attsortkeyord) as n_sortkeys,
max(attencodingtype) as max_enc
from pg_attribute group by 1) as det
on det.attrelid = a.id
inner join ( select tbl, max(mbytes)::decimal(32)/min(mbytes) as ratio
from (select tbl, trim(name) as name, slice, count(*) as mbytes
from svv_diskusage group by tbl, name, slice )
group by tbl, name ) as dist_ratio on a.id = dist_ratio.tbl
join ( select sum(capacity) as  total
from stv_partitions where part_begin=0 ) as part on 1=1
where mbytes is not null
order by  mbytes desc;
```

識別具有巢狀迴圈的查詢

下列查詢可識別已針對巢狀迴路記錄提醒事件的查詢。如需如何修正巢狀迴路條件的資訊，請參閱巢狀迴圈。

```
select query, trim(querytxt) as SQL, starttime
from stl_query
where query in (
select distinct query
from stl_alert_event_log
where event like 'Nested Loop Join in the query plan%')
order by starttime desc;
```

檢閱查詢的佇列等候時間

下列查詢顯示最近的查詢在執行之前等候查詢佇列中開放位置的時間。如果您看見較高的等候時間趨勢，您可能想要修改您的查詢佇列組態以獲得更好的傳輸量。如需詳細資訊，請參閱 實作手動 WLM。

```
select trim(database) as DB , w.query,
substring(q.querytxt, 1, 100) as querytxt,  w.queue_start_time,
w.service_class as class, w.slot_count as slots,
w.total_queue_time/1000000 as queue_seconds,
w.total_exec_time/1000000 exec_seconds, (w.total_queue_time+w.total_Exec_time)/1000000
 as total_seconds
from stl_wlm_query w
left join stl_query q on q.query = w.query and q.userid = w.userid
where w.queue_start_Time >= dateadd(day, -7, current_Date)
and w.total_queue_Time > 0  and w.userid >1
and q.starttime >= dateadd(day, -7, current_Date)
order by w.total_queue_time desc, w.queue_start_time desc limit 35;
```

依資料表檢閱查詢提醒

下列查詢可識別已為其記錄提醒事件的資料表，也可識別最常引發的提醒類型。

如果具有已識別資料表資料列的 minutes 值較高，請檢查資料表，以查看它是否需要例行維護，例如對它執行 ANALYZE 或 VACUUM。

如果資料列的 count 值較高但 table 值為 null，請對 STL_ALERT_EVENT_LOG 執行查詢，以取得關聯的 event 值，調查為何這麼常引發該提醒。

```
select trim(s.perm_table_name) as table,
(sum(abs(datediff(seconds, s.starttime, s.endtime)))/60)::numeric(24,0) as minutes,
 trim(split_part(l.event,':',1)) as event,  trim(l.solution) as solution,
max(l.query) as sample_query, count(*)
from stl_alert_event_log as l
left join stl_scan as s on s.query = l.query and s.slice = l.slice
and s.segment = l.segment and s.step = l.step
where l.event_time >=  dateadd(day, -7, current_Date)
group by 1,3,4
order by 2 desc,6 desc;
```

識別具有遺漏統計資訊的資料表

下列查詢提供您要對遺漏統計資料的資料表執行的查詢計數。如果此查詢傳回任何資料列，請查看 plannode 值來判斷受影響的資料表，然後對其執行ANALYZE。

```
select substring(trim(plannode),1,100) as plannode, count(*)
from stl_explain
where plannode like '%missing statistics%'
group by plannode
order by 2 desc;
```

對查詢進行故障診斷

此小節提供快速的參考，用於識別和解決使用 Amazon Redshift 查詢時可能遇到的部分最常見和最嚴重的問題。

主題

- 連線失敗
- 查詢懸置
- 查詢耗費太長時間
- 載入失敗
- 載入耗費太長時間
- 載入資料不正確
- 設定 JDBC 擷取大小參數

這些建議可做為進行故障排除的起點。您也可以參閱下列資源以取得更詳細的資訊。

- 存取 Amazon Redshift 叢集和資料庫
- 使用自動資料表最佳化
- 載入資料
- 教學課程：從 Amazon S3 載入資料

連線失敗

您的查詢連線可能因下列原因而失敗；建議您使用下列故障排除方法。

用戶端無法連線至伺服器

如果您使用 SSL 或伺服器憑證，對連線問題進行故障排除時，請先移除此複雜度。然後在您找到解決方案時，將 SSL 或伺服器憑證新增回去。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的設定連線的安全性選項。

連線遭拒

一般來說，收到錯誤訊息指出無法建立連線時，它表示存取叢集的許可發生問題。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的連線遭拒或失敗。

查詢懸置

您的查詢可能因下列原因而懸置或停止回應；建議您使用下列故障排除方法。

對資料庫的連線已丟棄

減少最大傳輸單位 (MTU) 的大小。MTU 大小決定可以透過您的網路連線在一個乙太網路訊框中傳輸的封包最大大小 (位元組)。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的已中斷與資料庫的連線。

對資料庫的連線逾時

執行長時間查詢 (例如 COPY 命令) 時，對資料庫的用戶端連線似乎懸置或逾時。在此情況下，您可能會發現 Amazon Redshift 主控台顯示查詢已完成，但用戶端工具本身仍似乎在執行查詢。取決於連線停止的時間，查詢的結果可能遺漏或不完整。當中繼網路元件終止閒置的連線時，會發生此影響。如需詳細資訊，請前往《Amazon Redshift 管理指南》中的防火牆逾時問題。

ODBC 發生用戶端 out-of-memory 錯誤

若您的用戶端應用程式使用 ODBC 連線，且您的查詢建立的結果集太大，無法納入記憶體中，則您可以使用資料指標將結果集串流到用戶端應用程式。如需詳細資訊，請參閱 DECLARE 及 使用游標時的效能考量。

JDBC 發生用戶端 out-of-memory 錯誤

當您嘗試透過 JDBC 連線擷取大型結果集時，可能會遇到用戶端 out-of-memory 錯誤。如需詳細資訊，請參閱 設定 JDBC 擷取大小參數。

可能有死鎖

如果有可能的死鎖，請嘗試下列：

- 檢視 STV_LOCKS 和 STL_TR_CONFLICT 系統資料表來尋找牽涉到對超過一個資料表進行更新的衝突。
- 使用 PG_CANCEL_BACKEND 函數來取消一或多個有衝突的查詢。
- 使用 PG_TERMINATE_BACKEND 函數來終止工作階段，它可強制終止的工作階段中任何目前執行中的交易釋出所有鎖定和復原交易。
- 排程並行寫入操作時務必謹慎。如需詳細資訊，請參閱 管理並行寫入操作。

查詢耗費太長時間

您的查詢連線可能因下列原因而耗費太長時間；建議您使用下列故障排除方法。

資料表未進行優化

設定資料表的排序索引鍵、配送樣式和壓縮編碼，以善加利用平行處理。如需更多資訊，請參閱 使用自動資料表最佳化

查詢正在寫入磁碟

您的查詢可能會在查詢執行的至少一部分寫入至磁碟。如需詳細資訊，請參閱 改善 查詢效能。

查詢必須等候其他查詢完成

您可以透過建立查詢佇列和指派不同類型的查詢至適當的佇列，來改善整體系統效能。如需詳細資訊，請參閱 實作工作負載管理。

查詢未進行優化

分析解釋計畫以尋找重新寫入查詢或最佳化資料庫的機會。如需詳細資訊，請參閱 查詢計劃。

查詢需要更多記憶體來執行

如果特定查詢需要更多記憶體，您可以透過增加wlm_query_slot_count來增加可用記憶體。

資料庫要求執行 VACUUM 命令

每當您新增、刪除或修改大量資料列時執行 VACUUM 命令，除非您以排序索引鍵順序載入您的資料。VACUUM 命令會重新組織您的資料以保有排序順序和還原效能。如需詳細資訊，請參閱 清空資料表。

疑難排解長期執行查詢的其他資源

以下是有助於調整查詢的系統檢視主題和其他文件章節：

- STV_INFLIGHT 系統檢視會顯示叢集上執行的查詢。將它與 STV_RECENTS 一起使用，以確定當前正在執行或最近完成的查詢會很有幫助。
- SYS_QUERY_HISTORY對於疑難排解很有用。它顯示 DDL 和 DML 查詢以及相關屬性，例如其目前的狀態 (例如 `running` 或 `failed`)、每個查詢執行所花費的時間，以及查詢是否在並行擴展叢集上執行。
- STL_QUERYTEXT 會擷取 SQL 命令的查詢文字。此外，將 STL_QUERYTEXT 聯結至 STV_INFLIGHT 的 SVV_QUERY_INFLIGHT，會顯示更多查詢中繼資料。
- 交易鎖定衝突可能是查詢效能問題的可能來源。如需目前在資料表上保留鎖定之交易的相關資訊，請參閱SVV_TRANSACTIONS。
- 識別最適合調整的查詢會提供疑難排解查詢，協助您判斷最近執行的查詢最耗時。這可以幫助您將精力集中在需要改進的查詢上。
- 如果您想要進一步探索查詢管理並瞭解如何管理查詢佇列，實作工作負載管理顯示如何執行此操作。工作負載管理是一項進階功能，我們建議您在大多數情況下自動化工作負載

載入失敗

您的資料載入可能因下列原因而失敗；建議您使用下列故障排除方法。

資料來源位於不同的 AWS 區域

根據預設，複製命令中指定的 Amazon S3 儲存貯體或 Amazon DynamoDB 表必須與叢集位於相同的 AWS 區域。如果您的資料和您的叢集位於不同的區域，您會收到類似以下的錯誤：

```
The bucket you are attempting to access must be addressed using the specified endpoint.
```

如果有可能，請確定您的叢集和您的資料來源位於相同區域。您可以使用 REGION 選項搭配 COPY 命令來指定不同的區域。

> **Note**
>
> 如果叢集和資料來源位於不同的 AWS 區域，則會產生資料傳輸費用。您的延遲也會更高。

COPY 命令失敗

查詢 STL_LOAD_ERRORS 以探索特定載入期間發生的錯誤。如需詳細資訊，請參閱 STL_LOAD_ERRORS。

載入耗費太長時間

您的載入操作可能因下列原因而耗費太長時間；建議您使用下列故障排除方法。

從單一檔案 COPY 載入資料

將您的載入資料分割至多個檔案。當您從多個檔案載入所有資料時，Amazon Redshift 將被迫執行序列化載入，其速度非常慢。檔案的數量應該是您的叢集中配量數量的倍數，而檔案應該是大約相等的大小，壓縮後介於 1 MB 與 1 GB 之間。如需詳細資訊，請參閱 設計查詢的 Amazon Redshift 最佳實務。

載入操作使用多個 COPY 命令

如果您同時使用多個 COPY 命令從多個檔案載入一個資料表，Amazon Redshift 將被迫執行序列化載入，其速度非常慢。在此情況下，請使用單一 COPY 命令。

載入資料不正確

您的 COPY 操作可能以下列方式載入不正確的資料；建議您使用下列故障排除方法。

載入錯誤的檔案

使用物件字首來指定資料檔案可能導致讀取不需要的檔案。相反地，請使用資訊清單檔案來指定要載入的確切檔案。如需詳細資訊，請參閱 COPY 命令的 copy_from_s3_manifest_file 選項和 COPY 範例中的 Example: COPY from Amazon S3 using a manifest。

設定 JDBC 擷取大小參數

依預設，JDBC 驅動程式會一次收集查詢的所有結果。因此，當您嘗試透過 JDBC 連線擷取大型結果集時，可能會遇到用戶端 out-of-memory 錯誤。若要讓用戶端能夠以批次方式擷取結果集，而非單一 all-or-nothing 擷取，請在用戶端應用程式中設定 JDBC fetch size 參數。

> **Note**
>
> ODBC 不支援擷取大小。

為求最佳效能，請將擷取大小設定為不會導致記憶體不足錯誤的最高值。較低的擷取大小值會造成更多伺服器來回行程，進而延長執行時間。伺服器會預留資源，包括 WLM 查詢位置和關聯的記憶體，直到用戶端擷取整個結果集或查詢取消為止。適當地調校擷取大小時，那些資源會更快速釋出，使得它們可供其他查詢使用。

> **Note**
>
> 如果您需要擷取大型資料集，建議您使用 UNLOAD 陳述式將資料傳輸到 Amazon S3。使用 UNLOAD 時，運算節點會平行運作，以加速資料的傳輸。

如需設定 JDBC 擷取大小參數的相關資訊，請前往 PostgreSQL 文件中的根據游標取得結果。

實作工作負載管理

您可以使用工作負載管理 (WLM) 來定義多個查詢佇列，並於執行時間將查詢路由至適當的佇列。

在某些情況下，您可能有多個工作階段或使用者在同時執行查詢。在這些情況下，某些查詢可能長時間佔用叢集資源，而影響其他查詢的效能。例如，假設有一群使用者偶而會提交複雜、長期執行的查詢，這些查詢會從數個大型資料表選取資料列並加以排序。另一群經常提交短期查詢，而這些查詢只從一個或兩個資料表選取少數幾列，且只會執行幾秒鐘。在此情況下，短期執行的查詢可能必須在佇列中等待長期執行的查詢完成。WLM 可協助管理這種情況。

您可以將 Amazon Redshift WLM 設定成以自動 WLM 或手動 WLM 來執行。

自動 WLM

若要最大化系統輸送量並有效使用資源，您可讓 Amazon Redshift 以自動 WLM 管理如何分配資源來執行並行查詢。自動 WLM 會管理執行查詢所需的資源。Amazon Redshift 會決定多少個查詢並行執行和分配多少記憶體給每個分派的查詢。您可以在 Amazon Redshift 主控台選擇切換 WLM 模式，然後選擇自動 WLM，以啟用自動 WLM。使用此選擇時，最多使用八個佇列來管理查詢，而且 Memory (記憶體) 和 Concurrency on main (主要叢集的並行) 欄位都設為 Auto (自動)。您可以指定優先順序來反映對應到每個佇列之工作負載或使用者的優先順序。查詢的預設優先順序設定為 Normal (一般)。如需如何變更佇列中查詢優先順序的相關資訊，請參閱查詢優先順序。如需詳細資訊，請參閱 實作自動 WLM。

在執行時間，您可以根據使用者群組或查詢群組，將查詢路由至這些佇列。您也可以設定一個查詢監控規則 (QMR) 來限制長時間執行查詢。

使用並行擴展和自動 WLM，您可以藉由持續的快速查詢效能，支援幾乎無限的並行使用者和並行查詢。如需詳細資訊，請參閱 使用並行擴展。

> **Note**
>
> 我們建議您建立參數群組並選擇自動 WLM 來管理查詢資源。如需如何從手動 WLM 遷移至自動 WLM 的詳細資訊，請參閱從手動 WLM 遷移至自動 WLM。

手動 WLM

或者，您可以修改 WLM 組態來為長期執行的查詢和短期執行的查詢建立個別的佇列，以管理系統效能和改善使用者的感受。在執行時間，您可以根據使用者群組或查詢群組，將查詢路由至這些佇列。您可

以在 Amazon Redshift 主控台切換到手動 WLM，以啟用此手動組態。使用此選擇時，請指定用來管理查詢的佇列，以及 Memory (記憶體) 和 Concurrency on main (主要叢集的並行) 欄位值。使用手動組態時，您最多可以設定八個查詢佇列，並設定每個佇列中可同時執行的查詢數。

您可以設定規則，以根據執行查詢的使用者，或您指定的標籤，將查詢路由至特定的佇列。您也可以設定要配置給每個佇列的記憶體數量，讓大型查詢在記憶體較多的佇列中執行。您也可以設定一個查詢監控規則 (QMR) 來限制長時間執行查詢。如需詳細資訊，請參閱 實作手動 WLM。

> **Note**
>
> 建議為您的手動 WLM 查詢佇列設定總共 15 個或更少的查詢槽。如需詳細資訊，請參閱 並行層級。

WLM 佇列限制

請注意，關於手動 WLM 組態，您可以分配給佇列的插槽上限為 50 個。不過，這並不表示在自動 WLM 組態中，Amazon Redshift 叢集一律會同時執行 50 個查詢。這可能會根據叢集上的記憶體需求或其他類型的資源配置而變更。

自動 WLM 和手動 WLM 的使用案例

當您希望 Amazon Redshift 管理資源分割方式以執行並行查詢時，請使用自動 WLM。使用自動 WLM 產生的輸送量通常會比手動 WLM 更高。使用自動 WLM，您可以為佇列中的工作負載定義查詢優先順序。如需查詢優先順序的相關資訊，請參閱 查詢優先順序。

當您想要對並行操作進行更多控制時，請使用手動 WLM。

主題

- 修改 WLM 組態
- 實作自動 WLM
- 實作手動 WLM
- 使用並行擴展
- 使用短期查詢加速
- WLM 佇列指派規則
- 將查詢指派給佇列
- WLM 動態和靜態組態屬性

- WLM 查詢監控規則
- WLM 系統資料表和檢視

修改 WLM 組態

修改 WLM 組態最簡單的方式是使用 Amazon Redshift 主控台。您也可以使用 AWS CLI 或 Amazon Redshift API。

在自動和手動 WLM 之間切換叢集時，您的叢集會進入 `pending reboot` 狀態。變更要在下次叢集重新啟動後才會生效。

如需修改 WLM 組態的詳細資訊，請參閱《Amazon Redshift 管理指南》中的設定工作負載管理。

從手動 WLM 遷移至自動 WLM

若要最大化系統輸送量並最有效地使用資源，我們建議您為佇列設定自動 WLM。請考慮採用以下方法來設定從手動 WLM 順暢過渡到自動 WLM。

若要從手動 WLM 遷移到自動 WLM 並使用查詢優先順序，我們建議您建立新的參數群組，然後將該參數群組附加到叢集。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 參數群組。

> **Important**
>
> 若要變更參數群組或從手動切換到自動 WLM，需要重新啟動叢集。如需詳細資訊，請參閱 WLM 動態和靜態組態屬性。

我們使用一個範例，其中有三個手動 WLM 佇列。一個用於 ETL 工作負載、一個用於分析工作負載，一個用於資料科學工作負載。ETL 工作負載每 6 小時執行一次，分析工作負載全天執行，資料科學工作負載可能隨時激增。使用手動 WLM，您可以根據對於每個工作負載對業務重要性的理解，來指定每個工作負載佇列可取得的記憶體和並行。指定記憶體和並行不僅很難釐清，也會導致叢集資源被靜態分割，因而在僅執行一部分工作負載時造成浪費。

您可以將自動 WLM 搭配查詢優先順序一起使用，來指示工作負載的相對優先順序，以避免上述問題。對於這個範例，請依照下列步驟進行：

- 建立一個新參數群組並切換到 Auto WLM (自動 WLM) 模式。

- 為三個工作負載各別新增佇列：ETL 工作負載、分析工作負載和資料科學工作負載。針對與 Manual WLM (手動 WLM) 模式搭配使用的每一個工作負載使用相同的使用者群組。
- 將 ETL 工作負載的優先順序指定為 `High`、分析工作負載的優先順序指定為 `Normal`、資料科學的優先順序指定為 `Low`。這些優先順序反映了不同工作負載或使用者群組的業務優先順序。
- 或者，您也可以為分析或資料科學佇列啟用並行擴展，以便在即使 ETL 工作負載每 6 小時執行一次時，這些佇列中的查詢仍能獲得一致的效能。

運用查詢優先順序，當只有分析工作負載在叢集上執行時，它可取得整個系統。這可以產生高輸送量和最佳的系統使用率 不過，當 ETL 工作負載啟動時，它會獲得正確的權限，因為它具有更高的優先順序。做為 ETL 工作負載執行的查詢，除了在被核准之後可獲得優先資源分配，也能優先獲得許可。因此，無論系統上有哪些其他項目正在執行，ETL 工作負載都能如預期地執行。高優先順序工作負載能獲得可預測的效能，代價是其他較低優先順序工作負載的執行時間更久，因為它們的查詢要等候較重要的查詢先完成；或者，當它們與較高優先順序的查詢同時執行時，所獲得的資源較少。Amazon Redshift 所用的排程演算法可使較低優先順序的查詢不會面臨資源耗盡，而是持續取得進展，只是速度較慢而已。

> **Note**
>
> - 逾時欄位不適用於自動 WLM。請改用 QMR 規則 `query_execution_time`。如需詳細資訊，請參閱 WLM 查詢監控規則。
> - QMR 動作 HOP 不適用於自動 WLM。請改用 `change priority` 動作。如需詳細資訊，請參閱 WLM 查詢監控規則。
> - 叢集使用自動 WLM 和手動 WLM 佇列的方式不同，這可能會使您的組態混淆。例如，您可以在自動 WLM 佇列中設定優先順序屬性，但不能在手動 WLM 佇列中設定優先順序屬性。因此，請避免在參數群組中混合使用自動 WLM 佇列和手動 WLM 佇列。而是在遷移到自動 WLM 時建立新的參數群組。

實作自動 WLM

自動工作負載管理 (WLM) 可讓 Amazon Redshift 管理查詢並行和記憶體配置。您可以使用服務類別識別碼 100-107 建立最多八個佇列。每個佇列都有優先順序。如需詳細資訊，請參閱 查詢優先順序。

自動 WLM 會決定查詢需要的資源數量，並根據工作負載來調整並行。當系統中有需要大量資源的查詢時 (例如，大型資料表之間的雜湊聯結)，並行較低。提出較輕型的查詢時 (例如，插入、刪除、掃描或簡易的彙總)，並行較高。

自動 WLM 不同於短期查詢加速 (SQA)，它以不同的方式評估查詢。即使長時間執行、非常消耗資源的查詢處於作用中狀態，將自動 WLM 和 SQA 搭配運作，也可以讓短期執行和輕量型查詢完成。如需 SQA 的相關資訊，請參閱使用短期查詢加速。

Amazon Redshift 透過參數群組啟用自動 WLM：

- 如果您的叢集使用預設參數群組，Amazon Redshift 可為其啟用自動 WLM。
- 如果您的叢集使用自訂參數群組，您可以設定叢集來啟用自動 WLM。我們建議您為自動 WLM 組態另外建立參數群組。

若要設定 WLM，請在可與叢集相關聯的參數群組中編輯 `wlm_json_configuration` 參數。如需詳細資訊，請參閱 修改 WLM 組態。

您在 WLM 組態中定義查詢佇列。您可以將更多查詢佇列新增至預設 WLM 組態，最多總共八個使用者佇列。您可以為每個查詢佇列設定下列項目：

- 優先順序
- 並行擴展模式
- 使用者群組
- 查詢群組
- 查詢監控規則

優先順序

您可以透過設定優先順序值來定義查詢在工作負載中的相對重要性。為佇列指定優先順序，與佇列關聯的所有查詢就會繼承其優先順序。如需詳細資訊，請參閱 查詢優先順序。

並行擴展模式

當並行擴展啟用時，當您需要更多叢集容量以執行增加的並行讀取及寫入查詢時，Amazon Redshift 將會自動新增額外的叢集容量。您的使用者會看到最新資料，無論查詢是執行於主要叢集或並行擴展叢集。

您可藉由設定 (WLM) 佇列來管理要將哪些查詢傳送至並行擴展叢集。當您為佇列啟用並行擴展時，合格查詢將傳送到並行擴展叢集，而非在佇列中等待。如需詳細資訊，請參閱 使用並行擴展。

使用者群組

您可以指定每一個使用者群組名稱或使用萬用字元，將一組使用者群組指派給佇列。當所列的使用者群組中有一個成員執行查詢時，該查詢會在相應佇列中執行。可指派給佇列的使用者群組數沒有固定的限制。如需詳細資訊，請參閱 根據使用者群組將查詢指派給佇列。

查詢群組

您可以指定每一個查詢群組名稱或使用萬用字元，將一組查詢群組指派給佇列。查詢群組只是一個標籤。在執行時間，您可以將查詢群組標籤指派給一系列查詢。指派給所列之查詢群組的任何查詢會在相應佇列中執行。可指派給佇列的查詢群組數沒有固定的限制。如需詳細資訊，請參閱 將查詢指派給查詢群組。

萬用字元

如果 WLM 佇列組態中啟用萬用字元，您可以個別地或利用 Unix shell 樣式的萬用字元，將使用者群組和查詢群組指派給佇列。模式比對不區分大小寫。

例如，'*' 萬用字元符合任意數目的字元。因此，如果您將 dba_* 新增至佇列的使用者群組清單，任何使用者執行查詢只要是屬於以 dba_ 為名稱開頭的群組，都指派到該佇列。例如 dba_admin 或 DBA_primary。'?' 萬用字元符合任何單一字元。因此，如果佇列包含使用者群組 dba?1，則名為 dba11 和 dba21 的使用者群組符合，但 dba12 不符。

預設情況下，不會啟用萬用字元。

查詢監控規則

查詢監控規則會為 WLM 佇列定義以指標為基礎的效能邊界，並指定當查詢超出這些邊界時採取的動作。例如，對於短期執行查詢專用的佇列，您可以建立規則，以將執行時間超過 60 秒的查詢取消。若要追蹤設計不良的查詢，您可以使用另一個規則來記錄含有巢狀迴圈的查詢。如需詳細資訊，請參閱 WLM 查詢監控規則。

檢查自動 WLM

若要檢查是否已啟用自動 WLM，請執行下列查詢。如果查詢傳回至少一列，表示自動 WLM 已啟用。

```
select * from stv_wlm_service_class_config
where service_class >= 100;
```

下列查詢顯示通過每個查詢佇列 (服務類別) 的查詢數。也顯示平均執行時間、第 90 個百分位數的查詢數和等待時間，以及平均等待時間。自動 WLM 查詢使用服務類別 100 到 107。

```
select final_state, service_class, count(*), avg(total_exec_time),
percentile_cont(0.9) within group (order by total_queue_time), avg(total_queue_time)
from stl_wlm_query where userid >= 100 group by 1,2 order by 2,1;
```

若要找出哪些查詢由自動 WLM 執行且成功完成，請執行下列查詢。

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class >= 100 and a.final_state = 'Completed'
order by b.query desc limit 5;
```

查詢優先順序

並非所有查詢都具有相同的重要性，通常某個工作負載或某一組使用者的效能會比較重要。如果您已啟用自動 WLM，則可透過設定優先順序值來定義查詢在工作負載中的相對重要性。為佇列指定優先順序，與佇列關聯的所有查詢就會繼承其優先順序。將使用者群組和查詢群組對應到佇列，即可將查詢關聯至佇列。您可以設定以下優先順序 (從最高優先順序到最低優先順序列出)：

1. HIGHEST
2. HIGH
3. NORMAL
4. LOW
5. LOWEST

當優先順序不同的查詢爭用相同資源時，管理員使用這些優先順序來顯示其工作負載的相對重要性。Amazon Redshift 會採用優先順序來將查詢放入系統，以及決定分配給查詢的資源量。依預設，查詢的優先順序設定為 NORMAL。

超級使用者可使用一個額外的優先順序 CRITICAL，這是比 HIGHEST 更高的優先順序。若要設定此優先順序，您可使用函數 CHANGE_QUERY_PRIORITY、CHANGE_SESSION_PRIORITY 和 CHANGE_USER_PRIORITY。若要將使用這些函數的許可授予給資料庫使用者，您可以建立預存程序並授予許可給使用者。如需範例，請參閱CHANGE_SESSION_PRIORITY。

> **Note**
>
> 一次只能執行一個 CRITICAL 查詢。

我們使用一個範例，其中擷取、轉換、載入 (ETL) 工作負載的優先順序高於分析工作負載的優先順序。ETL 工作負載每六個小時執行一次，而分析工作負載全天執行。當只有分析工作負載在叢集上執行時，它可取得整個系統，產生高輸送量和最佳的系統使用率。不過，當 ETL 工作負載啟動時，它會取得正確的權限，因為它具有更高的優先順序。做為 ETL 工作負載執行的查詢可優先獲得許可，在被核准之後也能獲得優先資源分配。因此，無論系統上有哪些其他項目正在執行，ETL 工作負載都能如預期地執行。因此，它能提供可預測的效能，以及讓管理員為其商業使用者提供服務水準協議 (SLA)。

在指定叢集中，高優先順序工作負載能獲得可預測的效能，代價是其他較低優先順序的工作負載。較低優先順序的工作負載可能會執行得更久，因為它們的查詢要等候較重要的查詢先完成。或者，當它們與較高優先順序的查詢同時執行時，所獲得的資源較少，所以要執行得更久。較低優先順序的查詢不會面臨資源耗盡，只是以較慢的速度取得進展。

在前面的範例中，管理員可以為分析工作負載啟用並行擴展。這麼做可以在即使 ETL 工作負載以高優先順序執行時，也能讓分析工作負載保持輸送量。

設定佇列優先順序

如果您已啟用自動 WLM，則每個佇列都具有優先順序值。查詢會根據使用者群組和查詢群組而路由到佇列。從將佇列優先順序設定為 NORMAL 開始。請根據與佇列之使用者群組和查詢群組關聯的工作負載，設定更高或更低的優先順序。

您可以在 Amazon Redshift 主控台上變更佇列的優先順序。在 Amazon Redshift 主控台上，工作負載管理頁面會顯示佇列並啟用佇列屬性的編輯，例如優先順序。若要使用 CLI 或 API 操作來設定優先順序，請使用 wlm_json_configuration 參數。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的設定工作負載管理。

下列 wlm_json_configuration 範例定義三個使用者群組 (ingest、reporting 和 analytics)。來自這些群組的使用者所提交的查詢分別以 highest、normal 和 low 的優先順序執行。

```
[
    {
        "user_group": [
            "ingest"
        ],
```

```
        "priority": "highest",
        "queue_type": "auto"
    },
    {
        "user_group": [
            "reporting"
        ],
        "priority": "normal",
        "queue_type": "auto"
    },
    {
        "user_group": [
            "analytics"
        ],
        "priority": "low",
        "queue_type": "auto",
        "auto_wlm": true
    }
]
```

使用查詢監控規則變更查詢優先順序

查詢監視規則 (QMR) 可讓您根據查詢執行時的行為，來變更查詢的優先順序。您可以在 QMR 述詞中指定優先順序屬性以及動作，來執行此作業。如需詳細資訊，請參閱 WLM 查詢監控規則。

例如，您可以定義規則，來取消任何分類為 high 優先順序且執行超過 10 分鐘的查詢。

```
"rules" :[
  {
    "rule_name":"rule_abort",
    "predicate":[
      {
        "metric_name":"query_cpu_time",
        "operator":">",
        "value":600
      },
      {
        "metric_name":"query_priority",
        "operator":"=",
        "value":"high"
      }
    ],
    "action":"abort"
```

```
  }
]
```

另一個範例是定義規則以將目前優先順序為 normal 且溢出超過 1 TB 到磁碟之任何查詢的優先順序變更為 lowest。

```
"rules":[
  {
    "rule_name":"rule_change_priority",
    "predicate":[
      {
        "metric_name":"query_temp_blocks_to_disk",
        "operator":">",
        "value":1000000
      },
      {
        "metric_name":"query_priority",
        "operator":"=",
        "value":"normal"
      }
    ],
    "action":"change_query_priority",
    "value":"lowest"
  }
]
```

監控查詢優先順序

若要顯示等待中和執行中查詢的優先順序，請檢視 stv_wlm_query_state 系統資料表中的 query_priority 欄。

```
query    | service_cl | wlm_start_time             | state            | queue_time |
 query_priority
---------+------------+----------------------------+------------------+------------
+----------------
2673299  | 102        | 2019-06-24 17:35:38.866356 | QueuedWaiting    | 265116     |
 Highest
2673236  | 101        | 2019-06-24 17:35:33.313854 | Running          | 0          |
 Highest
2673265  | 102        | 2019-06-24 17:35:33.523332 | Running          | 0          |
 High
```

```
2673284  | 102        | 2019-06-24 17:35:38.477366 | Running          | 0          |
 Highest
2673288  | 102        | 2019-06-24 17:35:38.621819 | Running          | 0          |
 Highest
2673310  | 103        | 2019-06-24 17:35:39.068513 | QueuedWaiting    | 62970      |
 High
2673303  | 102        | 2019-06-24 17:35:38.968921 | QueuedWaiting    | 162560     |
 Normal
2673306  | 104        | 2019-06-24 17:35:39.002733 | QueuedWaiting    | 128691     |
 Lowest
```

若要列出已完成查詢的優先順序，請檢視 stl_wlm_query 系統資料表中的 query_priority 欄。

```
select query, service_class as svclass, service_class_start_time as starttime,
 query_priority
from stl_wlm_query order by 3 desc limit 10;
```

```
  query  | svclass |         starttime          |    query_priority
---------+---------+----------------------------+----------------------
 2723254 |     100 | 2019-06-24 18:14:50.780094 | Normal
 2723251 |     102 | 2019-06-24 18:14:50.749961 | Highest
 2723246 |     102 | 2019-06-24 18:14:50.725275 | Highest
 2723244 |     103 | 2019-06-24 18:14:50.719241 | High
 2723243 |     101 | 2019-06-24 18:14:50.699325 | Low
 2723242 |     102 | 2019-06-24 18:14:50.692573 | Highest
 2723239 |     101 | 2019-06-24 18:14:50.668535 | Low
 2723237 |     102 | 2019-06-24 18:14:50.661918 | Highest
 2723236 |     102 | 2019-06-24 18:14:50.643636 | Highest
```

為了最佳化工作負載的輸送量，Amazon Redshift 可能會修改使用者所提交查詢的優先順序。Amazon Redshift 會使用進階機器學習演算法來判斷此最佳化何時有利於您的工作負載，並在符合下列所有條件時自動套用。

- 自動 WLM 已啟用。
- 僅會定義一個 WLM 佇列。
- 您尚未定義用來設定查詢優先順序的查詢監控規則 (QMR)。此類規則包括 QMR 指標 query_priority 或 QMR 動作 change_query_priority。如需詳細資訊，請參閱 WLM 查詢監控規則。

實作手動 WLM

使用手動 WLM，您可以修改 WLM 組態來為長期執行的查詢和短期執行的查詢建立個別的佇列，以管理系統效能和改善使用者的感受。

當使用者在 Amazon Redshift 中執行查詢時，會將查詢路由至查詢佇列。每個查詢佇列包含一些查詢槽。每個佇列都配置有叢集的一部分可用的記憶體。佇列的記憶體再分配給佇列的查詢槽。您可讓 Amazon Redshift 以自動 WLM 來管理查詢並行。如需詳細資訊，請參閱 實作自動 WLM。

或者，您可以設定每個查詢佇列的 WLM 屬性。這麼做可以指定各個槽分配記憶體的方式，以及在執行期將查詢路由至特定佇列的方式。您也可以設定 WLM 屬性來取消長時間執行的查詢。

根據預設，Amazon Redshift 會設定以下查詢佇列：

- 一個超級使用者佇列

  超級使用者佇列會保留給超級使用者專用，因此無法進行設定。只有在需要執行會影響系統的查詢，或基於故障診斷目的，才應該使用此佇列。例如，當您需要取消使用者長期執行的查詢，或需要將使用者新增至資料庫時，請使用此佇列。請勿用來執行例行性查詢。此佇列不會出現主控台，而是在資料庫的系統資料表中以第五個佇列出現。若要在超級使用者佇列中執行查詢，使用者必須以超級使用者身分登入，且必須使用預先定義的 `superuser` 查詢群組來執行查詢。
- 一個預設使用者佇列

  預設佇列最初會設定為同時執行五個查詢。當您使用手動 WLM 時，您可以變更預設佇列的並行、逾時和記憶體配置屬性，但無法指定使用者群組或查詢群組。預設佇列必須是 WLM 組態中的最後佇列。任何未路由至其他佇列的查詢會在預設佇列中執行。

查詢佇列定義於 WLM 組態。在一或多個可以與叢集相關聯的參數群組中，WLM 組態是可編輯的參數 (`wlm_json_configuration`)。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的設定工作負載管理。

您可以將更多查詢佇列新增至預設 WLM 組態，最多總共八個使用者佇列。您可以為每個查詢佇列設定下列項目：

- 並行擴展模式
- 並行層級
- 使用者群組
- 查詢群組

- 要使用的 WLM 記憶體百分比
- WLM 逾時
- WLM 查詢佇列跳轉
- 查詢監控規則

並行擴展模式

當並行擴展啟用時，當您需要更多叢集容量以執行增加的並行讀取及寫入查詢時，Amazon Redshift 將會自動新增額外的叢集容量。使用者會看到最新資料，無論查詢是執行於主要叢集或並行擴展叢集。

您可藉由設定 (WLM) 佇列來管理要將哪些查詢傳送至並行擴展叢集。當您為佇列啟用並行擴展時，合格查詢將傳送到並行擴展叢集，而非在佇列中等待。如需詳細資訊，請參閱 使用並行擴展。

並行層級

佇列中的查詢可以同時執行，直到達到該佇列已定義的 WLM 查詢槽計數 (或並行層級) 為止。後續的查詢需要在佇列中等待。

> **Note**
>
> WLM 並行層級不同於可對叢集建立的並行使用者連線數。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的連線到叢集。

在自動 WLM 組態中 (建議使用)，並行層級設為自動。Amazon Redshift 會動態地將記憶體配置給查詢，然後決定要同時執行多少個查詢。這是基於執行中和佇列查詢所需的資源。自動 WLM 無法設定。如需詳細資訊，請參閱 實作自動 WLM。

在手動 WLM 組態中，Amazon Redshift 會以靜態方式為每個佇列配置固定數量的記憶體。佇列的記憶體會在查詢插槽之間平均分割。說明如下，如果佇列分配到 20% 的叢集記憶體，並且具有 10 個插槽，則每個查詢都會分配到 2% 的叢集記憶體。無論同時執行的查詢數目為何，記憶體配置都會保持固定。因為此固定的記憶體配置，如果查詢在槽計數為 5 時可完全在記憶體中執行，當槽計數增加到 20 時，就可能將中間結果寫入磁碟。在這種情況下，每個查詢的佇列記憶體共用會從 1/5 減少到 1/20。增加的磁碟 I/O 會降低效能。

所有使用者定義佇列上的插槽計數上限為 50 個。這會限制所有佇列 (包括預設佇列) 的總插槽數。唯一不受限制限制的佇列是保留的超級使用者佇列。

根據預設，手動 WLM 佇列的並行層級為 5。在某些情況下，較高的並行層級可能有利於工作負載，例如：

- 如果許多較小的查詢被迫等待長期執行的查詢，請建立槽計數較高的另一個佇列，並將較小的查詢指派至該佇列。並行層級較高的佇列配置給每個查詢槽的記憶體較少，但較小的查詢所需的記憶體不多。

  > **Note**
  >
  > 如果您啟用短期查詢加速 (SQA)，WLM 會自動優先排定短期查詢，而長期執行查詢的優先順序會較低，所以您不需要為大部分工作流程的短期查詢準備另一個佇列。如需詳細資訊，請參閱 使用短期查詢加速。

- 如果您有多個查詢，每一個都會存取單一配量上的資料，請設定個別的 WLM 佇列以同時執行這些查詢。Amazon Redshift 會將並行查詢指派給不同配量，以便在多個配量上平行執行多個查詢。例如，假設查詢是在散發索引鍵上搭配述詞執行的簡單彙總，則查詢的資料會位於單一配量上。

一個手動 WLM 範例

此範例是一個簡單的手動 WLM 案例，用來說明插槽和記憶體的配置方式。您可以使用三個佇列來實作手動 WLM，這些佇列如下：

- 資料擷取佇列 – 這是針對擷取資料而設定的。這會分配 20% 的叢集記憶體，並且具有 5 個插槽。隨後，5 個查詢可以在佇列中同時運行，並且每個查詢會分配到 4% 的記憶體。
- 資料科學家佇列 - 這是專為記憶體密集型查詢而設計的。這會分配 40% 的叢集記憶體，並且具有 5 個插槽。隨後，5 個查詢可以同時運行，並且每個查詢會分配到 8% 的記憶體。
- 預設佇列 – 這是針對組織中大多數使用者所設計的。這包括銷售和會計群組，這些群組通常具有不複雜的簡短或中型執行查詢。它會分配到 40% 的叢集記憶體，並且具有 40 個插槽。40 個查詢可以在此佇列中同時執行，而每個查詢會分配到 1% 的記憶體。這是可配置給此佇列的插槽數目上限，因為所有佇列之間的限制為 50 個。

如果您正在執行自動 WLM，且工作負載需要並行執行 15 個以上的查詢，建議您開啟並行擴展。這是因為將查詢槽計數增加到 15 以上可能會導致系統資源爭用，並限制單一叢集的整體輸送量。使用並行擴展，您可以平行執行數百個查詢，最多可達設定的並行擴展叢集數目。並行擴展叢集數目由 max_concurrency_scaling_clusters 控制。如需並行擴展的相關資訊，請參閱 使用並行擴展。

如需詳細資訊，請參閱 改善 查詢效能。

使用者群組

您可以指定每一個使用者群組名稱或使用萬用字元，將一組使用者群組指派給佇列。當所列的使用者群組中有一個成員執行查詢時，該查詢會在相應佇列中執行。可指派給佇列的使用者群組數沒有固定的限制。如需詳細資訊，請參閱 根據使用者群組將查詢指派給佇列。

查詢群組

您可以指定每一個查詢群組名稱或使用萬用字元，將一組查詢群組指派給佇列。查詢群組只是一個標籤。在執行時間，您可以將查詢群組標籤指派給一系列查詢。指派給所列之查詢群組的任何查詢會在相應佇列中執行。可指派給佇列的查詢群組數沒有固定的限制。如需詳細資訊，請參閱 將查詢指派給查詢群組。

萬用字元

如果 WLM 佇列組態中啟用萬用字元，您可以個別地或利用 Unix shell 樣式的萬用字元，將使用者群組和查詢群組指派給佇列。模式比對不區分大小寫。

例如，'*' 萬用字元符合任意數目的字元。因此，如果您將 dba_* 新增至佇列的使用者群組清單，任何使用者執行查詢只要是屬於以 dba_ 為名稱開頭的群組，都指派到該佇列。例如 dba_admin 或 DBA_primary。'?' 萬用字元符合任何單一字元。因此，如果佇列包含使用者群組 dba?1，則名為 dba11 和 dba21 的使用者群組符合，但 dba12 不符。

依預設，萬用字元處於關閉狀態。

要使用的 WLM 記憶體百分比

在自動 WLM 組態中，記憶體百分比設為 **auto**。如需詳細資訊，請參閱 實作自動 WLM。

在手動 WLM 組態中，若要指定配置給查詢的可用記憶體數量，您可以設定 WLM Memory Percent to Use 參數。根據預設，每個使用者定義佇列都配置有相等份額的記憶體，可供使用者定義的查詢使用。例如，假設您有四個使用者定義佇列，則會配置可用記憶體的 25% 給每一個佇列。超級使用者佇列有其自己配置的記憶體，無法修改。若要變更配置，請將記憶體的整數百分比指派給每個佇列，總計最多 100%。任何未配置的記憶體由 Amazon Redshift 管理並可暫時提供給佇列 (如果佇列要求更多記憶體來進行處理)。

例如，假設您設定四個佇列，則可以如下配置記憶體：20%、30%、15%、15%。剩餘 20% 未配置，由服務管理。

WLM 逾時

WLM 逾時 (max_execution_time) 已作廢。這時請改成使用 query_execution_time 建立查詢監控規則 (QMR)，以限制查詢的經歷執行時間。如需詳細資訊，請參閱 WLM 查詢監控規則。

若要限制給定的 WLM 佇列中允許查詢使用的時間，您可以設定每個佇列的 WLM 逾時值。逾時參數指定 Amazon Redshift 在取消或轉跳查詢之前，等待查詢開始執行的時間量 (以毫秒為單位)。逾時以查詢執行時間為基礎，不包括在佇列中等待所花的時間。

WLM 會嘗試轉跳 CREATE TABLE AS (CTAS) 陳述式和唯讀查詢，例如 SELECT 陳述式。無法轉跳的查詢會取消。如需詳細資訊，請參閱 WLM 查詢佇列跳轉。

WLM 逾時不適用於已達到傳回狀態的查詢。若要檢視查詢的狀態，請參閱 STV_WLM_QUERY_STATE 系統資料表。COPY 陳述式和維護操作 (例如 ANALYZE 和 VACUUM) 不受制於 WLM 逾時。

WLM 逾時的功能類似於 statement_timeout 組態參數。差別在於 statement_timeout 組態參數適用於整個叢集，而 WLM 逾時是針對 WLM 組態中的單一佇列。

如果也指定 statement_timeout，則會使用 statement_timeout 和 WLM timeout (max_execution_time) 之中較小者。

查詢監控規則

查詢監控規則會為 WLM 佇列定義以指標為基礎的效能邊界，並指定當查詢超出這些邊界時採取的動作。例如，對於短期執行查詢專用的佇列，您可以建立規則，以將執行時間超過 60 秒的查詢取消。若要追蹤設計不良的查詢，您可以使用另一個規則來記錄含有巢狀迴圈的查詢。如需詳細資訊，請參閱 WLM 查詢監控規則。

WLM 查詢佇列跳轉

查詢可能因為 WLM 逾時或查詢監控規則 (QMR) 跳轉動作而跳轉。您只能在手動 WLM 配置中跳轉查詢。

當查詢跳轉時，WLM 會嘗試根據 WLM 佇列指派規則，將查詢路由至下一個相符的佇列。如果查詢不符合其他任何佇列定義，則會取消此查詢。不會指派給預設佇列。

WLM 逾時動作

下表摘要說明不同類型的查詢因為 WLM 逾時的行為。

查詢類型	動作
INSERT、UPDATE 和 DELETE	取消
使用者定義的函數 (UDF)	取消
UNLOAD	取消
COPY	繼續執行
維護操作	繼續執行
處於 `returning` 狀態的唯讀查詢	繼續執行
處於 `running` 狀態的唯讀查詢	重新指派或重新啟動
CREATE TABLE AS (CTAS)、SELECT INTO	重新指派或重新啟動

WLM 逾時佇列跳轉

WLM 會跳轉下列類型的查詢 (如果逾時)：

- WLM 狀態為 `running` 的唯讀查詢，例如 SELECT 陳述式。若要尋找查詢的 WLM 狀態，請檢視 STV_WLM_QUERY_STATE 系統資料表的 STATE 欄。
- CREATE TABLE AS (CTAS) 陳述式。WLM 佇列跳轉同時支援使用者定義和系統產生的 CTAS 陳述式。
- SELECT INTO 陳述式。

不受制於 WLM 逾時的查詢會繼續在原始佇列中執行，直到完成。下列類型的查詢不受制於 WLM 逾時：

- COPY 陳述式
- 維護操作，例如 ANALYZE 和 VACUUM
- WLM 狀態已達到 `returning` 的唯讀查詢，例如 SELECT 陳述式。若要尋找查詢的 WLM 狀態，請檢視 STV_WLM_QUERY_STATE 系統資料表的 STATE 欄。

不符合因 WLM 逾時而跳轉之資格的查詢，一旦逾時就會取消。下列類型的查詢不符合因 WLM 逾時而跳轉的資格：

- INSERT、UPDATE 和 DELETE 陳述式
- UNLOAD 陳述式
- 使用者定義的函數 (UDF)

WLM 逾時重新指派和重新啟動的查詢

當查詢跳轉且找不到相符的佇列時，就會取消此查詢。

當查詢跳轉且找到相符的佇列時，WLM 會嘗試將查詢重新指派給新佇列。如果無法重新指派查詢，則會在新佇列中重新啟動此查詢，如下所述。

只有在下列所有條件成立時，才會重新指派查詢：

- 找到相符的查詢。
- 新佇列有足夠可用的槽可執行查詢。如果 wlm_query_slot_count 參數設定的值大於 1，查詢可能需要多個槽。
- 新佇列可用的記憶體至少與查詢目前使用的記憶體一樣多。

如果重新指派查詢，此查詢會在新佇列中繼續執行。中間結果會保留下來，因此對於總執行時間影響極小。

如果無法重新指派查詢，則會取消查詢，並於新佇列中重新啟動查詢。將會刪除中間結果。查詢會在佇列中等待，然後在有足夠的槽可用時開始執行。

QMR 跳轉動作

下表摘要說明不同類型的查詢因為 QMR 跳轉動作的行為。

查詢類型	動作
COPY	繼續執行
維護操作	繼續執行
使用者定義的函數 (UDF)	繼續執行

查詢類型	動作
UNLOAD	重新指派或繼續執行
INSERT、UPDATE 和 DELETE	重新指派或繼續執行
處於 `returning` 狀態的唯讀查詢	重新指派或繼續執行
處於 `running` 狀態的唯讀查詢	重新指派或重新啟動
CREATE TABLE AS (CTAS)、SELECT INTO	重新指派或重新啟動

若要查明由 QMR 跳轉的查詢是否已重新指派、重新啟動或取消，請查詢 STL_WLM_RULE_ACTION 系統日誌資料表。

QMR 跳轉動作重新指派和重新啟動的查詢

當查詢跳轉且找不到相符的佇列時，就會取消此查詢。

當查詢跳轉且找到相符的佇列時，WLM 會嘗試將查詢重新指派給新佇列。如果無法重新指派查詢，此查詢會在新佇列中重新啟動，或在原始佇列中繼續執行，如下所述。

只有在下列所有條件成立時，才會重新指派查詢：

- 找到相符的查詢。
- 新佇列有足夠可用的槽可執行查詢。如果 wlm_query_slot_count 參數設定的值大於 1，查詢可能需要多個槽。
- 新佇列可用的記憶體至少與查詢目前使用的記憶體一樣多。

如果重新指派查詢，此查詢會在新佇列中繼續執行。中間結果會保留下來，因此對於總執行時間影響極小。

如果無法重新指派查詢，此查詢會重新啟動或在原始佇列中繼續執行。如果重新啟動查詢，則會取消查詢，並於新佇列中重新啟動查詢。將會刪除中間結果。查詢會在佇列中等待，然後在有足夠的槽可用時開始執行。

教學：設定手動工作負載管理 (WLM) 佇列

概觀

建議在 Amazon Redshift 中設定自動工作負載管理 (WLM)。如需自動 WLM 的相關資訊，請參閱實作工作負載管理。不過，如果您需要多個 WLM 佇列，此教學課程會逐步解說如何在 Amazon Redshift 中設定手動工作負載管理 (WLM)。您可以藉由設定手動 WLM 來改善叢集中的查詢效能和資源配置。

Amazon Redshift 將使用者查詢路由到佇列以進行處理。WLM 定義如何將這些查詢路由到佇列。Amazon Redshift 預設有兩個用於查詢的佇列：一個給超級使用者使用，一個給使用者使用。超級使用者佇列無法設定，且一次只能處理一個佇列。只有為了故障排除的目的才需要保留此佇列。使用者佇列一次最多可以處理五個查詢，但您可以視需要變更佇列的並行層級以改變設定。

如果有多個使用者對資料庫執行查詢，您可能會發現另一種設定更有效率。例如，如果某些使用者執行資源密集型操作 (如 VACUUM)，這些操作可能會對非密集型的查詢 (如報表) 造成負面影響。您可以考慮加入額外的佇列，並設定它們處理不同的工作負載。

預估時間：75 分鐘

評估成本：50 美分

必要條件

您需要 Amazon Redshift 叢集、範例 TICKIT 資料庫和 Amazon Redshift RSQL 用戶端工具。如果您還沒有進行這些設定，請前往《Amazon Redshift 入門指南》和 Amazon Redshift RSQL。

章節

- 第 1 節：了解預設佇列處理行為
- 第 2 節：修改 WLM 查詢佇列組態
- 第 3 節：根據使用者群組和查詢群組將查詢路由至佇列
- 第 4 節：使用 wlm_query_slot_count 暫時覆寫佇列中的並行層級
- 第 5 節：清理資源

第 1 節：了解預設佇列處理行為

在開始設定手動 WLM 之前，先了解 Amazon Redshift 中佇列處理的預設行為很有幫助。您將在本節建立兩個資料庫檢視，這些檢視會傳回多個系統資料表的資訊。然後，您將執行一些測試查詢，以查看預設情況下如何路由查詢。如需系統資料表的相關資訊，請參閱系統資料表和檢視參考。

步驟 1：建立 WLM_QUEUE_STATE_VW 檢視

在此步驟，您要建立名為 WLM_QUEUE_STATE_VW 的檢視。此檢視會回傳下列系統資料表的資訊。

- STV_WLM_CLASSIFICATION_CONFIG
- STV_WLM_SERVICE_CLASS_CONFIG
- STV_WLM_SERVICE_CLASS_STATE

您將在整個教學課程中使用此檢視來監控您變更 WLM 組態後佇列發生的事。下表說明 WLM_QUEUE_STATE_VW 檢視傳回的資料。

資料欄	描述
佇列	與表示佇列的資料列關聯的數量。佇列數量決定了佇列在資料庫中的順序。
description	此值描述佇列是僅用於某些使用者群組、某些查詢群組、或是所有類型的查詢。
slots	配置給佇列的槽數量。
mem	配置給佇列的記憶體數量 (單位為 MB/槽)。
max_execution_time	在終止查詢之前允許查詢執行的時間。
user_*	此值指示 WLM 組態是否允許使用萬元字元來比對使用者群組。
query_*	此值指示 WLM 組態是否允許使用萬元字元來比對查詢群組。
queued	佇列中正在等待處理的查詢數量。
執行中	目前執行中的查詢數。
executed	已執行的佇列數量。

建立 WLM_QUEUE_STATE_VW 檢視

1. 開啟 Amazon Redshift SQL 並連接到您的 TICKIT 範例資料庫。如果您沒有此資料庫，請參閱必要條件。

2. 執行下列查詢以建立 WLM_QUEUE_STATE_VW 檢視。

```
create view WLM_QUEUE_STATE_VW as
select (config.service_class-5) as queue
, trim (class.condition) as description
, config.num_query_tasks as slots
, config.query_working_mem as mem
, config.max_execution_time as max_time
, config.user_group_wild_card as "user_*"
, config.query_group_wild_card as "query_*"
, state.num_queued_queries queued
, state.num_executing_queries executing
, state.num_executed_queries executed
from
STV_WLM_CLASSIFICATION_CONFIG class,
STV_WLM_SERVICE_CLASS_CONFIG config,
STV_WLM_SERVICE_CLASS_STATE state
where
class.action_service_class = config.service_class
and class.action_service_class = state.service_class
and config.service_class > 4
order by config.service_class;
```

3. 執行下列查詢以查看檢視中包含的資訊。

```
select * from wlm_queue_state_vw;
```

以下是結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(querytype: any)	5	836	0	false	false	0	1	160

步驟 2：建立 WLM_QUERY_STATE_VW 檢視

在此步驟，您要建立名為 WLM_QUERY_STATE_VW 的檢視。此檢視會回傳 STV_WLM_QUERY_STATE 系統資料表的資訊。

您將在整個教學課程中使用此檢視來監控執行中的佇列。下表說明 WLM_QUERY_STATE_VW 檢視傳回的資料。

資料欄	描述
query	查詢 ID。
佇列	佇列數量。
slot_count	配置給查詢的槽數量。
start_time	查詢開始的時間。
state	查詢的狀態，例如 executing (執行中)。
queue_time	查詢在佇列中花費的時間 (微秒)。
exec_time	查詢已執行的微秒數。

建立 WLM_QUERY_STATE_VW 檢視

1. 在 RSQL 中，執行下列查詢以建立 WLM_QUERY_STATE_VW 檢視。

```
create view WLM_QUERY_STATE_VW as
select query, (service_class-5) as queue, slot_count, trim(wlm_start_time) as
 start_time, trim(state) as state, trim(queue_time) as queue_time, trim(exec_time) as
 exec_time
from stv_wlm_query_state;
```

2. 執行下列查詢以查看檢視中包含的資訊。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

query	queue	slot_count	start_time	state	queue_time	exec_time
1249	1	1	2014-09-24 22:19:16	Executing	0	516

步驟 3：執行測試查詢

在此步驟中，您將在 RSQL 中從多個連線執行查詢，然後查看系統資料表以判斷系統如何路由要處理的查詢。

您在這個步驟需要開啟兩個 RSQL 視窗：

- 在第 1 個 RSQL 視窗中，您將使用您在本教學課程中已建立的檢視，執行查詢來監視佇列和查詢的狀態。
- 在第 2 個 RSQL 視窗中，您將執行長時間執行的查詢來變更在第 1 個 RSQL 視窗中找到的結果。

執行測試查詢

1. 開啟兩個 RSQL 視窗。如果您已經開啟一個視窗，則只需再開啟第二個視窗。這兩個連線皆可使用相同的使用者帳戶。
2. 在第 1 個 RSQL 視窗中執行下列查詢。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

query	queue	slot_count	start_time	state	queue_time	exec_time
1258	1	1	2014-09-24 22:21:03	Executing	0	549

此查詢會傳回自我參考的結果。目前正在執行的查詢是此檢視中的 SELECT 陳述式。此檢視的查詢一律會至少傳回一個結果。比較此結果與在下一步驟中啟動的長時間執行查詢所產生的結果。

3. 在第 2 個 RSQL 視窗中，從 TICKIT 範本資料庫執行查詢。此查詢應該會執行大約一分鐘，以便您有時間瀏覽 WLM_QUEUE_STATE_VW 檢視的結果以及之前建立的WLM_QUERY_STATE_VW 檢視的結果。在某些情況下，您可能發覺查詢執行時間不夠讓您查詢這兩個檢視。在這些情況下，您可以在 `l.listid` 上提高篩選條件的值，讓它執行更久。

> **Note**
>
> 為了縮短查詢執行時間並改善系統效能，Amazon Redshift 會將特定查詢類型的結果快取在領導者節點的記憶體中。啟用結果快取後，後續的查詢會執行得更快。若要防止查詢執行過快，可停用目前工作階段的結果快取。

若要關閉目前工作階段的結果快取，將 enable_result_cache_for_session 參數設為 `off`，如下所示。

```
set enable_result_cache_for_session to off;
```

在第 2 個 RSQL 視窗中執行下列查詢。

```
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <
 100000;
```

4. 在第 1 個 RSQL 視窗中，查詢 WLM_QUEUE_STATE_VW 和 WLM_QUERY_STATE_VW，並比較其結果與先前的結果。

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

以下為結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(querytype: any)	5	836	0	false	false	0	2	163

query	queue	slot_count	start_time	state	queue_time	exec_time
1267	1	1	2014-09-24 22:22:30	Executing	0	684
1265	1	1	2014-09-24 22:22:26	Executing	0	4080859

請注意，您之前的查詢結果與此步驟的結果之間有以下差異：

- 現在 WLM_QUERY_STATE_VW 中有兩個資料列。一個結果是在此檢視上執行 SELECT 操作的自我參考查詢。第二個結果是上一步驟中長時間執行的查詢。
- WLM_QUEUE_STATE_VW 中的執行中欄位已從 1 增加到 2。此欄位值表示佇列中有兩個正在執行的查詢。
- 每次在佇列中執行查詢，executed 欄位都會遞增。

WLM_QUEUE_STATE_VW 檢視有助於取得佇列的整體來到以及每個佇列中正在處理的查詢數量。WLM_QUERY_STATE_VW 檢視則有助於取得目前正在執行的各個查詢的更詳細檢視。

第 2 節：修改 WLM 查詢佇列組態

現在您已了解佇列的預設工作方式，接下來可學習如何使用手動 WLM 來設定查詢佇列。在本節中，您將為叢集建立及設定新的參數群組。您會建立另外兩個使用者佇列，並根據查詢的使用者群組或查詢群組標籤，將佇列設為接受查詢。任何未路由到這兩個佇列之一的查詢，都會在執行時路由到預設佇列。

在參數群組中建立手動 WLM 組態

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單中，選擇組態，然後選擇工作負載管理以顯示工作負載管理頁面。
3. 選擇 Create (建立) 以顯示 Create parameter group (建立參數群組) 視窗。
4. 在參數群組名稱和描述中輸入 **WLMTutorial**，然後選擇建立以建立參數群組。

   > **Note**
   >
   > Parameter group name (參數群組名稱) 會在建立時轉換成全部小寫的格式。

5. 在 Workload management (工作負載管理) 頁面上，選擇 **wlmtutorial** 參數群組來顯示詳細資訊頁面，其中包含 Parameters (參數) 和 Workload management (工作負載管理) 的標籤。
6. 確認您正在 Workload management (工作負載管理) 標籤上，然後選擇 Switch WLM mode (切換 WLM 模式) 來顯示 Concurrency settings (並行設定) 視窗。
7. 選擇 Manual WLM (手動 WLM)，然後選擇 Save (儲存) 來切換至手動 WLM。
8. 選擇 Edit workload queues (編輯工作負載佇列)。
9. 選擇 Add queue (新增佇列) 兩次來新增兩個佇列。現在有三個佇列：Queue 1 (佇列 1)、Queue 2 (佇列 2) 和 Default queue (預設佇列)。
10. 輸入每個佇列的資訊如下：

    - 對於佇列 1，在記憶體 (%)輸入 **30**、在主體的並行輸入 **2**，以及在查詢群組輸入 **test**。將其他設定保留為其預設值。
    - 對於佇列 2，在記憶體 (%)輸入 **40**、在主體的並行輸入 **3**，以及在使用者群組輸入 **admin**。將其他設定保留為其預設值。
    - 不要對 Default queue (預設佇列) 做任何變更。WLM 會將未配置的記憶體指派給預設佇列。

11. 選擇 Save (儲存) 儲存設定。

接下來，將具備手動 WLM 組態的參數群組與叢集建立關聯。

將具備手動 WLM 組態的參數群組與叢集建立關聯

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽選單上，選擇叢集，然後選擇叢集來顯示您叢集的清單。

3. 選擇您的叢集，例如 examplecluster，以顯示叢集的詳細資訊。然後選擇屬性索引標籤以顯示該叢集的屬性。
4. 在資料庫組態區段中，選擇編輯、編輯參數群組，以顯示參數群組視窗。
5. 對於參數群組，請選擇先前建立的 **wlmtutorial** 參數群組。
6. 選擇儲存變更以關聯參數群組。

   叢集會使用變更的參數群組進行修改。但是，您需要重新啟動叢集，才能將變更套用到資料庫。
7. 選擇您的叢集，然後在動作選擇重新啟動。

叢集重新啟動之後，狀態會回到 Available (可用)。

第 3 節：根據使用者群組和查詢群組將查詢路由至佇列

現在，您已將叢集與新的參數群組建立關聯，也已設定 WLM。接下來，執行一些查詢，以了解 Amazon Redshift 如何將查詢路由傳送到佇列來處理。

步驟 1：檢視資料庫中的查詢佇列組態

首先，確認資料庫有您預期的 WLM 組態。

檢視查詢佇列組態

1. 開啟 RSQL 並執行下列查詢。此查詢使用您在步驟 1：建立 WLM_QUEUE_STATE_VW 檢視中建立的 WLM_QUEUE_STATE_VW 檢視。如果叢集在重新啟動之前有已連線到資料庫的工作階段，則需要重新連線。

   ```
   select * from wlm_queue_state_vw;
   ```

   以下是結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	0	0	0
2	(user group: admin)	3	557	0	false	false	0	0	0
3	(querytype: any)	5	250	0	false	false	0	1	0

   比較這些結果與您在步驟 1：建立 WLM_QUEUE_STATE_VW 檢視收到的結果。請注意，現在有兩個額外的佇列。佇列 1 現在是 test 查詢群組的佇列，佇列 2 是 admin 使用者群組的佇列。

   佇列 3 現在是預設佇列。清單中的最後一個佇列一定是預設佇列。如果查詢未指定使用者群組或查詢群組，依預設會將查詢路由傳送到此佇列。

2. 執行下列查詢來確認您的查詢現在是在佇列 3 執行。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

query	queue	slot_count	start_time	state	queue_time	exec_time
2144	3	1	2014-09-24 23:49:59	Executing	0	550430

步驟 2：使用查詢群組佇列執行查詢

使用查詢群組佇列執行查詢

1. 執行以下查詢將其路由到 `test` 查詢群組。

```
set query_group to test;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. 在另一個 RSQL 視窗中執行下列查詢。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

query	queue	slot_count	start_time	state	queue_time	exec_time
2168	1	1	2014-09-24 23:54:18	Executing	0	6343309
2170	3	1	2014-09-24 23:54:24	Executing	0	847

查詢已路由到測試查詢群組，即現在的佇列 1。

3. 選取佇列狀態檢視中的所有項目。

```
select * from wlm_queue_state_vw;
```

您會看到類似以下的結果。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	0	1	0
2	(user group: admin)	3	557	0	false	false	0	0	0
3	(querytype: any)	5	250	0	false	false	0	1	3

4. 現在，重設查詢群組並再次執行長時間查詢：

```
reset query_group;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

5. 對檢視執行查詢以查看結果。

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

以下為結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	0	0	1
2	(user group: admin)	3	557	0	false	false	0	0	0
3	(querytype: any)	5	250	0	false	false	0	2	5

query	queue	slot_count	start_time	state	queue_time	exec_time
2186	3	1	2014-09-24 23:57:52	Executing	0	649
2184	3	1	2014-09-24 23:57:48	Executing	0	4137349

結果應該是查詢現在再次在佇列 3 中執行。

步驟 3：建立資料庫使用者和群組

您必須先在資料庫中建立使用者群組並在群組中新增使用者，才可以在此佇列中執行查詢。接著，您將使用新使用者的憑證登入 RSQL，然後執行查詢。您需要以超級使用者的身分 (例如管理員使用者) 執行查詢，才能建立資料庫使用者。

建立新的資料庫使用者和使用者群組

1. 在 RSQL 視窗中執行以下命令，可在資料庫中建立新的資料庫使用者，名為 adminwlm。

```
create user adminwlm createuser password '123Admin';
```

2. 然後執行下列命令來建立新使用者群組，並將您的新使用者 adminwlm 加入群組。

```
create group admin;
alter group admin add user adminwlm;
```

步驟 4：使用使用者群組佇列執行查詢

接下來您要執行查詢，並將其路由到使用者群組佇列。當您要將查詢路由到您設定用來處理想執行之查詢類型的佇列，請這麼做。

使用使用者群組佇列執行查詢

1. 在第 2 個 RSQL 視窗中，執行以下查詢切換到 adminwlm 帳戶，然後以該使用者的身分執行查詢。

```
set session authorization 'adminwlm';
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. 在第 1 個 RSQL 視窗中，執行以下查詢來看看查詢被路由到哪個查詢佇列。

```
select * from wlm_query_state_vw;
select * from wlm_queue_state_vw;
```

   以下為結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	0	0	1
2	(user group: admin)	3	557	0	false	false	0	1	0
3	(querytype: any)	5	250	0	false	false	0	1	8

query	queue	slot_count	start_time	state	queue_time	exec_time
2202	2	1	2014-09-25 00:01:38	Executing	0	4885796
2204	3	1	2014-09-25 00:01:43	Executing	0	650

   執行此查詢的佇列是佇列 2，即 admin 使用者佇列。每次您以此使用者身分登入並執行查詢時，這些查詢就會在佇列 2 中執行，除非您指定要使用其他查詢群組。所選佇列取決於佇列指派規則。如需詳細資訊，請參閱 WLM 佇列指派規則。

3. 現在，從第 2 個 RSQL 視窗執行下列查詢。

```
set query_group to test;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

4. 在第 1 個 RSQL 視窗中，執行以下查詢來看看查詢被路由到哪個查詢佇列。

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

以下為結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	0	1	1
2	(user group: admin)	3	557	0	false	false	0	0	1
3	(querytype: any)	5	250	0	false	false	0	1	10

query	queue	slot_count	start_time	state	queue_time	exec_time
2218	1	1	2014-09-25 00:04:30	Executing	0	4819666
2220	3	1	2014-09-25 00:04:35	Executing	0	685

5. 完成後，請重設查詢群組。

```
reset query_group;
```

第 4 節：使用 wlm_query_slot_count 暫時覆寫佇列中的並行層級

有時候使用者可能暫時需要較多資源來處理特定查詢。如果是這種情況，他們可以使用 wlm_query_slot_count 組態設定來暫時覆寫查詢佇列配置槽的方式。槽是用於處理查詢的記憶體和 CPU 的單位。當您有偶爾會佔用叢集中大量資源的查詢，例如在資料庫中執行 VACUUM 操作，可以覆蓋槽計數。

您可能發現使用者經常需要為某些類型的查詢設定 wlm_query_slot_count。若是如此，請考慮調整 WLM 組態，並提供更符合查詢需求的佇列給使用者。如需使用槽計數暫時覆寫並行層級的相關資訊，請參閱 wlm_query_slot_count。

步驟 1：使用 wlm_query_slot_count 覆寫並行層級

基於本教學課程的目的，我們將執行相同的長時間執行 SELECT 查詢。我們會以 adminwlm 使用者的身分執行，使用 wlm_query_slot_count 增加查詢可使用的槽數量。

使用 wlm_query_slot_count 覆寫並行層級

1. 增加查詢限制量，以確保您有足夠的時間查詢 WLM_QUERY_STATE_VW 檢視圖並查看結果。

```
set wlm_query_slot_count to 3;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. 現在，使用管理員使用者查詢 WLM_QUERY_STATE_VW 來查看查詢的執行狀況。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

query	queue	slot_count	start_time	state	queue_time	exec_time
2240	2	3	2014-09-25 00:08:45	Executing	0	3731414
2242	3	1	2014-09-25 00:08:49	Executing	0	596

請注意，查詢的槽計數是 3。此計數表示查詢會將三個槽都用來處理查詢，將佇列中的所有資源配置給該查詢。

3. 現在，執行下列查詢。

```
select * from WLM_QUEUE_STATE_VW;
```

以下是結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	0	0	4
2	(user group: admin)	3	557	0	false	false	0	1	3
3	(querytype: any)	5	250	0	false	false	0	1	25

wlm_query_slot_count 組態的設定只在目前的工作階段中有效。如果工作階段過期，或是有其他使用者執行查詢，則會使用 WLM 組態。

4. 重設槽計數，然後重新執行測試。

```
reset wlm_query_slot_count;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

以下為結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	0	0	2
2	(user group: admin)	3	557	0	false	false	0	1	2
3	(querytype: any)	5	250	0	false	false	0	1	14

query	queue	slot_count	start_time	state	queue_time	exec_time
2260	2	1	2014-09-25 00:12:11	Executing	0	4042618
2262	3	1	2014-09-25 00:12:15	Executing	0	680

步驟 2：從不同工作階段執行查詢

接下來，請從不同工作階段執行查詢。

從不同工作階段執行查詢

1. 在第 1 和第 2 個 RSQL 視窗中，執行下列查詢以使用 test 查詢群組。

```
set query_group to test;
```

2. 在第 1 個 RSQL 視窗中執行下列長時間查詢。

```
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

3. 由於長時間執行的查詢仍在第 1 個 RSQL 視窗中進行，請執行下列命令。這些命令會提高槽計數來使用佇列的所有槽，然後開始執行長時間執行的查詢。

```
set wlm_query_slot_count to 2;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

4. 開啟第 3 個 RSQL 視窗，查詢檢視並查看結果。

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

以下為結果範例。

queue	description	slots	mem	max_time	user_*	query_*	queued	executing	executed
0	(super user) and (query group: superuser)	1	357	0	false	false	0	0	0
1	(query group: test)	2	627	0	false	false	1	1	2
2	(user group: admin)	3	557	0	false	false	0	0	3
3	(querytype: any)	5	250	0	false	false	0	1	18

query	queue	slot_count	start_time	state	queue_time	exec_time
2286	1	2	2014-09-25 00:16:48	QueuedWaiting	3758950	0
2282	1	1	2014-09-25 00:16:33	Executing	0	19335850
2288	3	1	2014-09-25 00:16:52	Executing	0	666

請注意，第一個查詢使用已配置給佇列 1 的其中一個槽來執行查詢。此外，請注意有一個查詢在佇列中等待 (其中，queued 是 1，state 是 QueuedWaiting)。第一個查詢完成後，第二個查詢就會開始執行。發生此執行是因為兩個查詢都路由到 test 查詢群組，第二個查詢必須等到有足夠的槽才能開始處理。

第 5 節：清理資源

叢集只要執行就會繼續產生費用。完成本教學課程後，請按照《Amazon Redshift 入門指南》中的尋找其他資源和重設環境中的步驟，將環境恢復到先前的狀態。

如需 WLM 的相關資訊，請參閱實作工作負載管理。

使用並行擴展

使用並行擴展功能，您可以藉由持續的快速查詢效能，支援數千個並行使用者和並行查詢。開啟並行擴展時，Amazon Redshift 會自動新增額外的叢集容量以處理增加的讀取及寫入查詢。使用者會看到最新資料，無論查詢是執行於主要叢集或並行擴展叢集。

您可藉由設定 (WLM) 佇列來管理要將哪些查詢傳送至並行擴展叢集。當您開啟並行擴展時，合格查詢會傳送到並行擴展叢集，而非在佇列中等待。

我們只會向您收取並行擴展叢集實際執行查詢的費用。如需定價的相關資訊，包括費用應計方式與最低費用，請參閱並行擴展定價。

並行擴展功能

當您為 WLM 佇列開啟並行擴展時，其適用於讀取操作，例如儀表板查詢。也適用於常用的寫入操作，例如用於資料擷取和處理的陳述式。

寫入操作的並行擴展功能

並行擴展支援常用的寫入操作，例如擷取、轉換和載入 (ETL) 陳述式。當您想要在叢集收到大量要求時維持一致的回應時間時，寫入操作的並行擴展特別有用。這會改善在主叢集上爭用資源的寫入操作輸送量。

並行擴展支援 COPY、INSERT、DELETE、UPDATE 和 CREATE TABLE AS (CTAS) 陳述式。此外，並行擴展支援不使用彙總功能之 MV 的具體化視觀表重新整理。不支援其他資料操作語言 (DML) 陳述式和資料定義語言 (DDL) 陳述式。如果不支援的寫入陳述式 (例如沒有 TABLE AS 的 CREATE) 包含在支援的寫入陳述式之前的明確交易中，則不會在並行擴展叢集上執行任何寫入陳述式。

當您為並行擴展累積點數時，此點數累積適用於讀取與寫入操作。

並行擴展的限制

以下是使用 Amazon Redshift 並行擴展的限制：

- 不支援對使用交錯排序索引鍵的資料表進行查詢。
- 不支援對暫存資料表進行查詢。
- 不支援可存取受限制網路或虛擬私有雲端 (VPC) 組態保護的外部資源的查詢。
- 不支援包含 Python 使用者定義函數 (UDF) 和 Lambda UDF 的查詢。
- 不支援可存取系統資料表、PostgreSQL 目錄資料表或無備份資料表的查詢。
- 當有限制的 IAM 政策許可時，它不支援存取外部資源的 COPY 或 UNLOAD 查詢。這包括套用至資源 (例如 Amazon S3 儲存貯體或 DynamoDB 表) 的許可，或套用至來源的許可。IAM 來源可以包括下列項目：
  - `aws:sourceVpc`— 來源虛 VPC。
  - `aws:sourceVpce`— 來源虛 VPC 端點。
  - `aws:sourceIp`— 來源 IP 位址。

  在某些情況下，您可能需要移除限制資源或來源的權限，以便將存取資源的 COPY 和 UNLOAD 查詢傳送至並行擴展叢集。

  如需有關資源策略的詳細資訊，請參閱 AWS Identity and Access Management 使用指南中的策略類型和使用儲存貯體策略控制來自 VPC 端點的存取。
- DDL 作業 (例如 CREATE TABLE 或 ALTER TABLE) 不支援用於寫入操作的 Amazon Redshift 並行擴展。
- 不支援 COPY 命令的 ANALYZE。
- 不支援在 DISTSTYLE 設定為 ALL 的目標資料表上進行寫入操作。
- 不支援以下檔案格式的 COPY：
  - Parquet
  - ORC
- 不支援對具有身分資料欄的資料表進行寫入操作。
- Amazon Redshift 只在 Amazon Redshift RA3 節點上支援寫入操作的並行擴展，特別是 ra3.16xlarge、ra3.4xlarge 和 ra3.xlplus。其他節點類型不支援寫入操作的並行擴展。

AWS 區域 用於並行縮放

並行擴展可在下列 AWS 區域中使用：

- 美國東部 (維吉尼亞北部) 區域 (us-east-1)
- 美國東部 (俄亥俄) 區域 (us-east-2)

- AWS GovCloud (美國東部)
- 美國西部 (加利佛尼亞北部) 區域 (us-west-1)
- 美國西部 (奧勒岡) 區域 (us-west-2)
- 亞太區域 (孟買) 區域 (ap-south-1)
- 亞太區域 (首爾) 區域 (ap-northeast-2)
- 亞太區域 (新加坡) 區域 (ap-southeast-1)
- 亞太區域 (雪梨) 區域 (ap-southeast-2)
- 亞太區域 (東京) 區域 (ap-northeast-1)
- 加拿大 (中部) 區域 (ca-central-1)
- 歐洲 (法蘭克福) 區域 (eu-central-1)
- 歐洲 (愛爾蘭) 區域 (eu-west-1)
- 歐洲 (倫敦) 區域 (eu-west-2)
- 歐洲 (巴黎) 區域 (eu-west-3)
- 歐洲 (斯德哥爾摩) 區域 (eu-north-1)
- 南美洲 (聖保羅) 區域 (sa-east-1)

並行擴展候選項目

只有在主要叢集符合以下要求時，查詢才會路由至並行擴展叢集：

- EC2-VPC 平台。
- 節點類型必須是 DC2.8 x 大、DC2、ra3.XL 加號、大型或 ra3.16 倍大。僅下列 Amazon Redshift RA3 節點支援寫入操作的並行擴展：ra3.16xlarge、ra3.4xlarge 和 ra3.xlplus。
- 最多可容納 32 個運算節點，適用於具有 ra3.xlplus、ra3.4 x 大型節點或大型節點類型的叢集。此外，最初建立叢集時，主要叢集的節點數不得超過 32 個節點。例如，如果叢集最初建立時為 40 個節點，即使叢集目前具備 20 個節點，也不符合並行擴展的需求。相反地，如果 DC2 叢集目前有 40 個節點，但最初是使用 20 個節點建立，則它確實符合並行擴展的需求。
- 非單一節點叢集。

設定並行擴展佇列

您可以藉由將工作負載管理 (WLM) 佇列啟用為並行擴展佇列，來將查詢路由至並行擴展叢集。若要開啟佇列的並行擴展，請將 Concurrency Scaling mode (並行擴展模式) 值設定為 auto (自動)。

當路由至並行擴展佇列的查詢數量超過佇列設定的並行時，合格查詢將傳送到並行擴展叢集。當有可用的空位時，查詢將會執行於主要叢集。佇列數量僅受限於每個叢集允許的佇列數量。與任何 WLM 佇列相同，您可以依據使用者群組或以查詢群組標籤來標記查詢，以便將查詢路由至並行擴展佇列。您也可以藉由定義 WLM 查詢監控規則 來路由查詢。例如，您可以將耗時超過 5 秒的所有查詢路由至並行擴展佇列。

並行擴展叢集的預設數目為一。可使用的並行擴展叢集數目由 max_concurrency_scaling_clusters 控制。

監控並行擴展

若要查看查詢是執行於主要叢集或並行擴展叢集，您可以瀏覽至 Amazon Redshift 主控台中的叢集，然後選擇一個叢集。然後選擇查詢監控索引標籤和工作負載並行，檢視有關執行中查詢和佇列查詢的資訊。

若要找出執行時間，請查詢 STL_QUERY 資料表並篩選 concurrency_scaling_status 欄。以下查詢可針對執行於並行擴展叢集與執行於主要叢集的查詢，比較其佇列時間與執行時間。

```
SELECT w.service_class AS queue
, CASE WHEN q.concurrency_scaling_status = 1 THEN 'concurrency scaling cluster' ELSE
 'main cluster' END as concurrency_scaling_status
, COUNT( * ) AS queries
, SUM( q.aborted ) AS aborted
, SUM( ROUND( total_queue_time::NUMERIC / 1000000,2) ) AS queue_secs
, SUM( ROUND( total_exec_time::NUMERIC / 1000000,2) ) AS exec_secs
FROM stl_query q
JOIN stl_wlm_query w
USING (userid,query)
WHERE q.userid > 1
AND q.starttime > '2019-01-04 16:38:00'
AND q.endtime < '2019-01-04 17:40:00'
GROUP BY 1,2
ORDER BY 1,2;
```

根據您的需求調整 starttime 和 endtime 值。

並行擴展系統檢視

有一組字首為 SVCS 的系統檢視可提供來自系統日誌資料表的詳細資訊，內容是有關主要叢集與並行擴展叢集上的查詢。

以下檢視提供與對應的 STL 檢視或 SVL 檢視類似的資訊：

- SVCS_ALERT_EVENT_LOG
- SVCS_COMPILE
- SVCS_EXPLAIN
- SVCS_PLAN_INFO
- SVCS_QUERY_SUMMARY
- SVCS_STREAM_SEGS

以下檢視專屬於並行擴展。

- SVCS_CONCURRENCY_SCALING_USAGE

如需並行擴展的相關資訊，請參閱《Amazon Redshift 管理指南》中的下列主題。

- 檢視並行擴展資料
- 檢視查詢執行期間的叢集效能
- 檢視查詢詳細資訊

使用短期查詢加速

短期查詢加速 (SQA) 可排定讓短期執行的查詢優先於長期執行的查詢。SQA 會在專用空間中執行短期查詢，所以 SQA 查詢不會被迫在佇列中排在長期查詢後面等待。SQA 只優先處理短期執行且位於使用者定義佇列中的查詢。SQA 可讓短期執行的查詢更快開始執行，使用者會更快看到結果。

如果啟用 SQA，您可以減少短期查詢執行專用的工作負載管理 (WLM) 佇列。此外，長期執行的查詢不需要與短期查詢爭奪佇列中的槽，因此，您可以將 WLM 佇列設定為使用較少的查詢槽。使用較低的並行性時，就大部分工作負載而言，查詢傳輸量會增加，且整體系統效能會改善。

CREATE TABLE AS (CTAS) 陳述式和唯讀查詢 (例如 SELECT 陳述式) 符合 SQA 的資格。

Amazon Redshift 採用機器學習演算法來分析每一個合格查詢，並預測查詢的執行時間。根據預設，WLM 會根據叢集工作負載的分析結果，動態指派 SQA 最長執行時間的值。或者，您可以指定固定值 1–20 秒。如果查詢的預測執行時間小於定義或動態指派的 SQA 最大執行期，而且查詢正在 WLM 佇列中等候，則 SQA 會將查詢與 WLM 佇列區隔，並排定其優先執行順序。如果查詢執行的時間比

SQA 最長執行時間還久，WLM 會根據 WLM 佇列指派規則，將查詢移至第一個相符的 WLM 佇列。隨著 SQA 從查詢模式中學習一段時間，預測會越準確。

根據預設，SQA 在預設參數群組中會啟用，且適用於所有新的參數群組。若要在 Amazon Redshift 主控台停用 SQA，請編輯參數群組的 WLM 組態，並取消選取啟用短期查詢加速。在最佳作法上，建議使用的 WLM 查詢槽計數為 15 或更少，以維持最佳的整體系統效能。如需修改 WLM 組態的資訊，請參閱《Amazon Redshift 管理指南》中的設定工作負載管理。

短期查詢最長執行時間

啟用 SQA 時，根據預設，WLM 會將短期查詢的最長執行時間設為動態。建議維持 SQA 最長執行時間的動態設定。您可以指定固定值 1–20 秒以覆寫預設設定。

在某些情況下，您可能考慮針對 SQA 最長執行時間值使用不同的值，以改善系統效能。在此情況下，請分析工作負載，以找出大部分短期執行查詢的最長執行時間。下列查詢傳回第 70 個百分位數附近之查詢的最長執行時間。

```
select least(greatest(percentile_cont(0.7)
within group (order by total_exec_time / 1000000) + 2, 2), 20)
from stl_wlm_query
where userid >= 100
and final_state = 'Completed';
```

決定適合工作負載的最長執行時間值之後，就不需要變更，除非工作負載大幅改變。

監控 SQA

若要檢查是否已啟用 SQA，請執行下列查詢。如果查詢傳回一列，表示 SQA 已啟用。

```
select * from stv_wlm_service_class_config
where service_class = 14;
```

下列查詢顯示通過每個查詢佇列 (服務類別) 的查詢數。也顯示平均執行時間、第 90 個百分位數的查詢數和等待時間，以及平均等待時間。SQA 查詢使用服務類別 14。

```
select final_state, service_class, count(*), avg(total_exec_time),
percentile_cont(0.9) within group (order by total_queue_time), avg(total_queue_time)
from stl_wlm_query where userid >= 100 group by 1,2 order by 2,1;
```

若要找出哪些查詢由 SQA 挑出且成功完成，請執行下列查詢。

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class = 14 and a.final_state = 'Completed'
order by b.query desc limit 5;
```

若要找出由 SQA 挑出但逾時的查詢，請執行下列查詢。

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class = 14 and a.final_state = 'Evicted'
order by b.query desc limit 5;
```

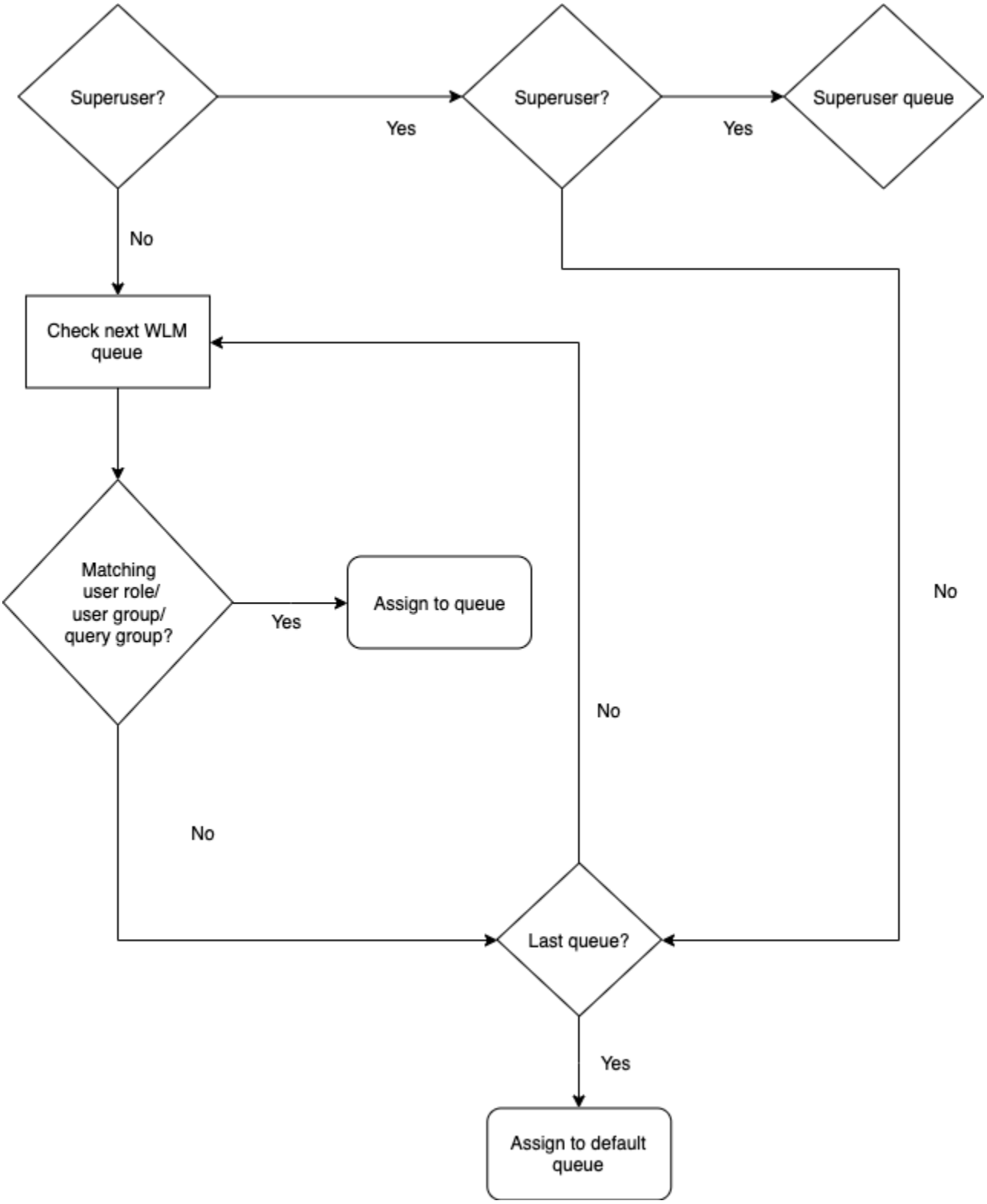
如需移出查詢的相關資訊，以及可對查詢執行之更普遍的規則式動作的詳細資訊，請參閱 WLM 查詢監控規則。

WLM 佇列指派規則

當使用者執行查詢時，WLM 會根據 WLM 佇列指派規則，將查詢指派給第一個相符的佇列：

1. 如果使用者以超級使用者身分登入，且在標示為超級使用者的查詢群組中執行查詢，此查詢會指派給超級使用者佇列。
2. 如果使用者是角色的一部份、屬於列出的使用者群組，或在列出的查詢群組內執行查詢，此查詢會指派給第一個相符的佇列。
3. 如果查詢不符合任何條件，此查詢會指派給預設佇列，此為 WLM 組態中定義的最後一個佇列。

下圖說明這些規則如何運作。

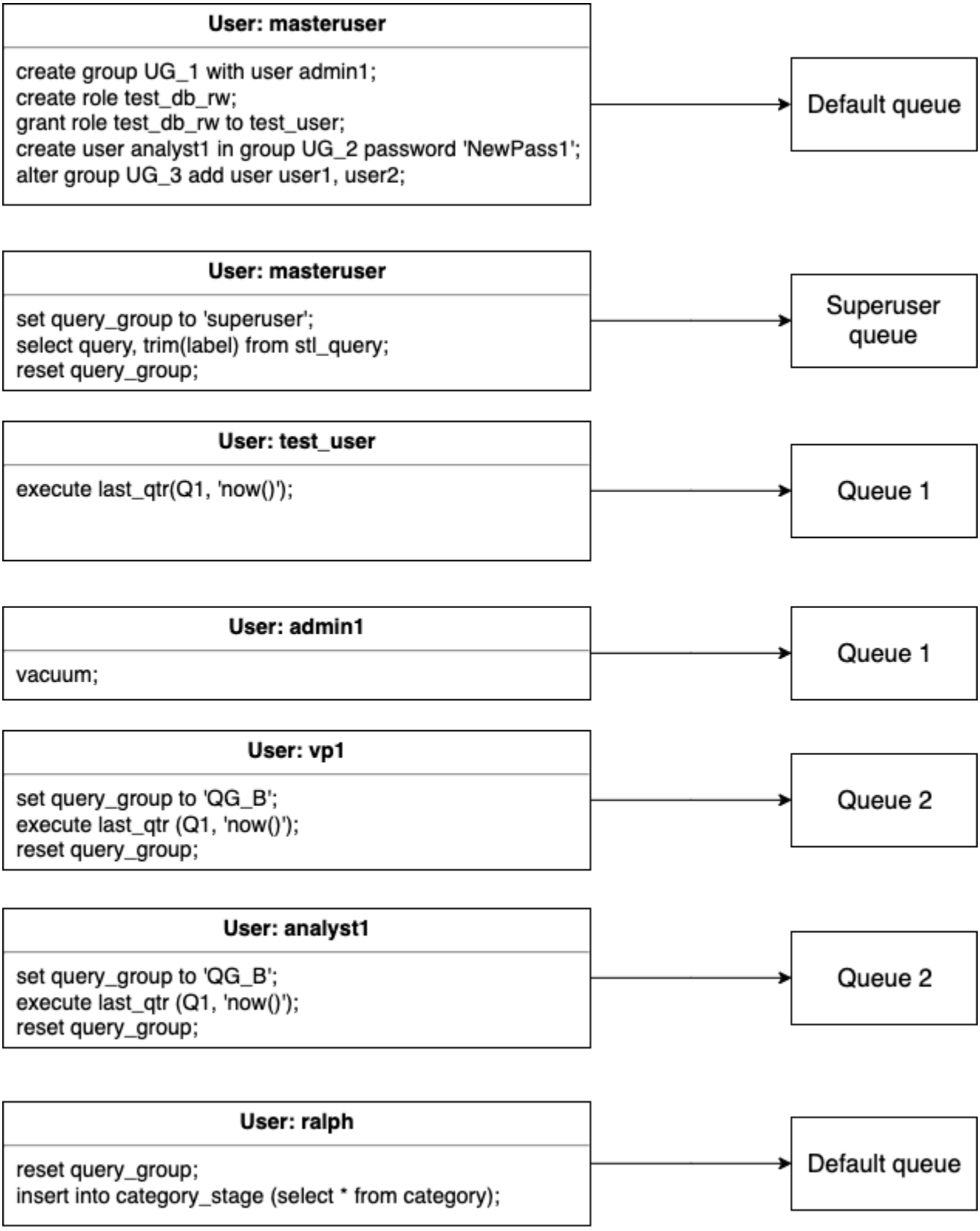


佇列指派範例

下表示範具有超級使用者佇列和四個使用者定義佇列的 WLM 組態。

佇列	並行數量	使用者角色	User Groups (使用者群組)	Query Groups (查詢群組)
超級使用者	1			superuser
1	5	test_db_rw	UG_1	
2	5			QG_B
3	5		UG_2	QG_C
預設	5			

下圖示範如何根據使用者群組和查詢群組，將查詢指派給上表中的佇列。如需在執行時間如何將查詢指派給使用者群組和查詢群組的資訊，請參閱本節稍後的將查詢指派給佇列。



在此範例中，WLM 執行下列指派：

1. 第一組陳述式示範將使用者指派給使用者群組的三種方法。陳述式由使用者 adminuser 執行，此使用者不是任何 WLM 佇列中所列出之使用者群組的成員。未設定任何查詢群組，所以會將陳述式路由至預設佇列。
2. 使用者 adminuser 是超級使用者，且查詢群組設為 'superuser'，所以會將查詢指派給超級使用者佇列。
3. 使用者 test_user 會獲派佇列 1 所列出的角色 test_db_rw，所以查詢會指派給佇列 1。
4. 使用者 admin1 是佇列 1 所列出之使用者群組的成員，所以查詢會指派給佇列 1。
5. 使用者 vp1 不是任何列出之使用者群組的成員。查詢群組設為 'QG_B'，所以會將查詢指派給佇列 2。
6. 使用者 analyst1 是佇列 3 中所列出之使用者群組的成員，但 'QG_B' 符合佇列 2，所以會將此查詢指派給佇列 2。
7. 使用者 ralph 不是任何列出之使用者群組的成員，且該查詢群組已重設，所以沒有相符的佇列。會將該查詢指派給預設佇列。

將查詢指派給佇列

下列範例會根據使用者角色、使用者群組和查詢群組，將查詢指派給佇列。

根據使用者角色將查詢指派給佇列

如果將使用者指派給某個角色，且該角色已附加至佇列，則該使用者執行的查詢會指派至該佇列。下列範例會建立名為 sales_rw 的使用者角色，並將使用者 test_user 指派給該角色。

```
create role sales_rw;
grant role sales_rw to test_user;
```

您也可以將一個角色明確授予另一個角色，來合併兩個角色的許可。將巢狀角色指派給使用者，會將這兩個角色的許可授予使用者。

```
create role sales_rw;
create role sales_ro;
grant role sales_ro to role sales_rw;
grant role sales_rw to test_user;
```

若要查看叢集中已被授予角色的使用者清單，請查詢 SVV_USER_GRANTS 資料表。若要查看叢集中已被授予角色的角色清單，請查詢 SVV_ROLE_GRANTS 資料表。

```
select * from svv_user_grants;
select * from svv_role_grants;
```

根據使用者群組將查詢指派給佇列

如果使用者群組名稱列在佇列定義中，則由該使用者群組的成員所執行的查詢會指派給相應佇列。下列範例使用 SQL 命令 CREATE USER、CREATE GROUP 和 ALTER GROUP，建立使用者群組並將使用者新增至群組。

```
create group admin_group with user admin246, admin135, sec555;
create user vp1234 in group ad_hoc_group password 'vpPass1234';
alter group admin_group add user analyst44, analyst45, analyst46;
```

將查詢指派給查詢群組

您可以在執行時間將查詢指派給適當的查詢群組，以便將查詢指派給佇列。使用 SET 命令來開始查詢群組。

```
SET query_group TO group_label
```

首先，*group_label* 是 WLM 組態中列出的查詢群組標籤。

您在 SET query_group 命令之後執行的所有查詢，將會以指定之查詢群組的成員身分執行，直到您重設查詢群組或結束目前的登入工作階段為止。如需有關設定和重設 Amazon Redshift 物件的資訊，請參閱 SQL 命令參考中的 SET 和 RESET。

您指定的查詢群組標籤必須包含在目前的 WLM 組態中；否則，SET query_group 命令對查詢佇列沒有作用。

TO 子句中定義的標籤會擷取到查詢日誌中，您可以利用此標籤來進行故障診斷。如需查詢群組組態參數的相關資訊，請參閱組態參考中的 query_group。

下列範例在查詢群組 'priority' 中執行兩個查詢，然後重設查詢群組。

```
set query_group to 'priority';
select count(*)from stv_blocklist;
select query, elapsed, substring from svl_qlog order by query desc limit 5;
reset query_group;
```

將查詢指派給超級使用者佇列

若要將查詢指派給超級使用者佇列，請以超級使用者身分登入 Amazon Redshift，然後在超級使用者群組中執行查詢。完成時，請重設查詢群組，後續的查詢就不會在超級使用者佇列中執行。

以下範例指派兩個命令在超級使用者佇列中執行。

```
set query_group to 'superuser';

analyze;
vacuum;
reset query_group;
```

若要檢視超級使用者清單，請查詢 PG_USER 系統目錄資料表。

```
select * from pg_user where usesuper = 'true';
```

WLM 動態和靜態組態屬性

WLM 屬性分為動態和靜態兩種。您可在不重新啟動叢集的情形下，將動態屬性套用至資料庫，但靜態屬性需要重新啟動叢集才能讓變更生效。但假如您同時變更了動態與靜態屬性，則必須重新啟動叢集，所有屬性變更才會生效。無論變更的屬性是動態或是靜態，都是如此。

套用動態屬性時，叢集狀態為 `modifying`。在自動 WLM 與手動 WLM 之間切換為靜態變更，需叢集重新啟動以生效。

下表指出使用自動 WLM 或手動 WLM 時，哪些 WLM 屬性是動態的，哪些是靜態的。

WLM 屬性	自動 WLM	手動 WLM
查詢群組	動態	靜態
查詢群組萬用字元	動態	靜態
使用者群組	動態	靜態
使用者群組萬用字元	動態	靜態
使用者角色	動態	靜態

WLM 屬性	自動 WLM	手動 WLM
使用者角色萬用字元	動態	靜態
主要叢集的並行	不適用	動態
並行擴展模式	動態	動態
啟用短期查詢加速	不適用	動態
短期查詢最長執行時間	動態	動態
要使用的記憶體百分比	不適用	動態
逾時	不適用	動態
優先順序	動態	不適用
新增或移除佇列	動態	靜態

如果您修改查詢監控規則 (QMR)，則變更會自動進行，而不需要修改叢集。

> **Note**
>
> 使用手動 WLM 時，如果變更逾時值，新值將套用至變更值之後才開始執行的任何查詢。如果變更要使用的記憶體並行或百分比，Amazon Redshift 將動態變更至新組態。如此，目前執行中的查詢將不會受變更影響。如需詳細資訊，請參閱 WLM 動態記憶體配置。

主題

- WLM 動態記憶體配置
- 動態 WLM 範例

WLM 動態記憶體配置

在每一個佇列中，WLM 會建立一些適合佇列並行層級的查詢槽。配置給查詢槽的記憶體數量等於配置給佇列的記憶體百分比除以槽計數。如果您變更記憶體配置或並行，Amazon Redshift 會動態設

法轉移至新的 WLM 組態。因此，作用中查詢可以使用目前配置的記憶體數量，執行到完成為止。同時，Amazon Redshift 會確保記憶體總用量絕不超過 100% 可用的記憶體。

工作負載管理員使用下列程序來管理轉移：

1. WLM 會重新計算每個新查詢槽的記憶體配置。
2. 如果執行中的查詢並沒有主動使用某個查詢槽，WLM 會移除此槽，而該記憶體就可以給新的槽使用。
3. 如果查詢槽目前在使用中，WLM 會等待查詢完成。
4. 當作用中查詢完成時，就會移除空的槽，並釋放相關的記憶體。
5. 只要有足夠的記憶體可供新增一或多個槽，就會新增新的槽。
6. 當變更時執行的所有查詢都完成時，槽計數就等於新的並行層級，而轉移至新的 WLM 組態也就完成。

事實上，變更生效時正在執行的查詢會繼續使用原始的記憶體配置。變更生效時已排入佇列的查詢會在有新的槽可用時，路由傳送至新的槽。

如果 WLM 動態屬性在轉移過程中變更，WLM 會從目前狀態開始，立即開始轉移至新的組態。若要檢視轉移的狀態，請查詢 STV_WLM_SERVICE_CLASS_CONFIG 系統資料表。

動態 WLM 範例

假設您的叢集 WLM 使用下列動態屬性設定兩個佇列。

佇列	並行數量	要使用的記憶體 %
1	4	50%
2	4	50%

現在，假設您的叢集有 200 GB 的記憶體可供查詢處理使用。(此為任意數字，僅供示範用途。) 如下列方程式所示，每個槽配置 25 GB。

```
(200 GB * 50% ) / 4 slots  = 25 GB
```

接下來，您將 WLM 變更為使用下列動態屬性。

佇列	並行數量	要使用的記憶體 %
1	3	75%
2	4	25%

如下列方程式所示，佇列 1 中每個槽的新記憶體配置為 50 GB。

```
(200 GB * 75% ) / 3 slots = 50 GB
```

假設套用新組態時，查詢 A1、A2、A3 和 A4 在執行中，而查詢 B1、B2、B3 和 B4 會排入佇列。WLM 會動態重新設定查詢槽，如下所示。

步驟	執行中的查詢	目前槽計數	目標槽計數	配置的記憶體	可用的記憶體量
1	A1、A2、A3、A4	4	0	100 GB	50 GB
2	A2、A3、A4	3	0	75 GB	75 GB
3	A3、A4	2	0	50 GB	100 GB
4	A3、A4、B1	2	1	100 GB	50 GB
5	A4、B1	1	1	75 GB	75 GB
6	A4、B1、B2	1	2	125 GB	25 GB
7	B1、B2	0	2	100 GB	50 GB
8	B1、B2、B3	0	3	150 GB	0 GB

1. WLM 會重新計算每個查詢槽的記憶體配置。佇列 1 最初配置 100 GB。新的佇列總共配置 150 GB，所以新佇列立即有 50 GB 可用。佇列 1 現在使用四個槽，且新的並行層級是三個槽，所以不會新增任何槽。

2. 當一個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有三個槽和 75 GB 的可用記憶體。新的組態規定每個新的槽要有 50 GB，但新的並行層級是三個槽，所以不會新增任何槽。
3. 當第二個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有兩個槽和 100 GB 的可用記憶體。
4. 使用 50 GB 的可用記憶體新增一個槽。佇列 1 現在有三個槽和 50 GB 的可用記憶體。排入佇列的查詢現在可以路由至新的槽。
5. 當第三個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有兩個槽和 75 GB 的可用記憶體。
6. 使用 50 GB 的可用記憶體新增一個槽。佇列 1 現在有三個槽和 25 GB 的可用記憶體。排入佇列的查詢現在可以路由至新的槽。
7. 當第四個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有兩個槽和 50 GB 的可用記憶體。
8. 使用 50 GB 的可用記憶體新增一個槽。佇列 1 現在有三個槽，各有 50 GB，所有可用的記憶體都已配置。

轉移完成，所有查詢槽都可供排入佇列的查詢使用。

WLM 查詢監控規則

在 Amazon Redshift 工作負載管理 (WLM) 中，查詢監控規則會為 WLM 佇列定義以指標為基礎的效能邊界，並指定當查詢超出這些邊界時要採取的動作。例如，對於短期執行查詢專用的佇列，您可以建立規則，以將執行時間超過 60 秒的查詢取消。若要追蹤設計不良的查詢，您可以使用另一個規則來記錄含有巢狀迴圈的查詢。

請將查詢監控規則定義為工作負載管理 (WLM) 組態的一部分。每一個佇列最多可以定義 25 個規則，而全部佇列合計以 25 個規則為限制。每個規則最多包含三個條件 (或述詞) 和一個動作。述詞由指標、比較條件 (=、< 或 >) 和值所組成。如果符合任何規則的所有述詞，就會觸發該規則的動作。可能的規則動作包括記錄、跳轉和中止，如下所述。

給定佇列中的規則僅套用至該佇列中執行的查詢。各規則彼此無關。

WLM 每 10 秒評估一次指標。如果同一時段內觸發多個規則，WLM 會啟動最嚴重的動作，依序為中止、跳轉、記錄。如果動作是跳轉或中止，則會記錄動作，並從佇列中移出查詢。如果動作是記錄，則查詢會在佇列中繼續執行。WLM 會依每個規則，對每個查詢只啟動一個記錄動作。如果佇列包含其他規則，那些規則仍然有效。如果動作是跳轉，且將查詢路由至另一個佇列，則會套用新佇列的規則。如需針對特定查詢執行的查詢監控和追蹤動作的相關資訊，請參閱 使用短期查詢加速 中的範例集合。

當符合規則的所有述詞時，WLM 會將一列寫入 STL_WLM_RULE_ACTION 系統資料表。此外，Amazon Redshift 會將目前執行中查詢的查詢指標記錄到 STV_QUERY_METRICS。已完成之查詢的指標儲存在 STL_QUERY_METRICS 中。

定義查詢監控規則

您可以將查詢監控規則定義為 WLM 組態的一部分，而 WLM 組態是您在叢集的參數群組定義中定義。

您可以使用 AWS Management Console 或以程式設計方式使用 JSON 來建立規則。

> **Note**
>
> 如果您選擇以程式設計方式建立規則，強烈建議您使用主控台來產生 JSON，再加入參數群組定義中。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的使用主控台建立或修改查詢監控規則和使用 AWS CLI設定參數值。

若要定義查詢監控規則，請指定下列元素：

- 規則名稱 – 規則名稱在 WLM 組態內必須是唯一的。規則名稱最多為 32 個英數字元或底線，且不可含有空格和問號。每個佇列最多可以有 25 個規則，而全部佇列合計最多 25 規則。
- 一或多個述詞 – 每個規則最多可以有三個述詞。如果符合任何規則的所有述詞，就會觸發相關聯的動作。述詞由指標名稱、運算子 (=、< 或 >) 和值所定義。例如，`query_cpu_time > 100000`。如需指標清單及各種指標的範例值，請參閱本節後續的已佈建 Amazon Redshift 的查詢監控指標。
- 動作 – 如果觸發多個規則，WLM 會選擇具有最嚴重動作的規則。依嚴重性遞增順序，可能的動作如下：
  - 日誌 – 將查詢的相關資訊記錄在 STL_WLM_RULE_ACTION 系統資料表中。只需要寫入日誌記錄時，請使用「記錄」動作。WLM 依每個規則，對每個查詢最多只建立一個日誌。在記錄動作之後，其他規則仍然有效，且 WLM 會繼續監控查詢。
  - 跳轉 (僅適用於手動 WLM) – 記錄動作，並將查詢跳轉至下一個相符的佇列。如果沒有另一個相符的佇列，則會取消查詢。QMR 只會轉跳 CREATE TABLE AS (CTAS) 陳述式和唯讀查詢，例如 SELECT 陳述式。如需詳細資訊，請參閱 WLM 查詢佇列跳轉。
  - 中止 – 記錄動作並終止查詢。QMR 不會停止 COPY 陳述式和維護操作，例如 ANALYZE 和 VACUUM。
  - 變更優先順序 (僅適用於自動 WLM) – 變更查詢的優先順序。

如要限制查詢的執行時間，建議您建立查詢監控規則，而不要使用 WLM 逾時。例如，您可將 `max_execution_time` 設定為 50,000 毫秒，如下列 JSON 程式碼片段所示。

```
"max_execution_time": 50000
```

但我們建議您改為定義對等查詢監控規則，將 query_execution_time 設定為 50 秒，如下列 JSON 程式碼片段所示。

```
"rules":
[
    {
        "rule_name": "rule_query_execution",
        "predicate": [
            {
                "metric_name": "query_execution_time",
                "operator": ">",
                "value": 50
            }
        ],
        "action": "abort"
    }
]
```

如需建立或修改查詢監控規則的步驟，請參閱《Amazon Redshift 管理指南》中的使用主控台建立或修改查詢監控規則和 wlm_json_configuration 參數中的屬性。

在下列主題中，您可以找到查詢監控規則的詳細資訊：

- 已佈建 Amazon Redshift 的查詢監控指標
- 查詢監控規則範本
- 使用主控台建立規則
- 設定工作負載管理
- 查詢監控規則的系統資料表和檢視

已佈建 Amazon Redshift 的查詢監控指標

下表描述查詢監控規則中使用的指標。(這些指標不同於 STV_QUERY_METRICS 和 STL_QUERY_METRICS 系統資料表中儲存的指標。)

對於給定的指標，將會在查詢層級或區段層級追蹤效能臨界值。如需區段和步驟的相關資訊，請參閱查詢計劃和執行工作流程。

> **Note**
>
> WLM 逾時 參數不同於查詢監控規則。

指標	名稱	描述
查詢 CPU 時間	query_cpu_time	查詢所用的 CPU 時間 (秒)。CPU time 不同於 Query execution time。 有效值為 0–999,999。
已讀取的區塊	query_blocks_read	查詢所讀取的 1 MB 資料區塊數。 有效值為 0–1,048,575。
掃描的資料列數	scan_row_count	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。 有效值為 0–999,999,999,999,999。
查詢執行時間	query_execution_time	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。 有效值為 0–86,399。
查詢佇列時間	query_queue_time	在佇列中等待的時間 (秒)。 有效值為 0–86,399。
CPU 用量	query_cpu_usage_percent	查詢所用的 CPU 容量百分比。 有效值為 0–6,399。
記憶體到磁碟	query_temp_blocks_to_disk	用來寫入中間結果的暫時磁碟空間，以 1 MB 區塊為單位。 有效值為 0–319,815,679。

指標	名稱	描述
CPU 扭曲	cpu_skew	任何分割 CPU 用量與所有分割之平均 CPU 用量的比率。此指標在區段層級中定義。 有效值為 0–99。
I/O 扭曲	io_skew	任何分割讀取區塊數上限 (I/O) 與所有分割平均讀取區塊數的比率。此指標在區段層級中定義。 有效值為 0–99。
已聯結的資料列	join_row_count	聯結步驟中處理的資料列數。 有效值為 0–999,999,999,999,999。
巢狀迴圈聯結資料列數	nested_loop_join_row_count	巢狀迴圈聯結中的數字或資料列。 有效值為 0–999,999,999,999,999。
傳回的資料列數	return_row_count	查詢傳回的資料列數。 有效值為 0–999,999,999,999,999。
區段執行時間	segment_execution_time	單一區段經過的執行時間 (秒)。為了避免或減少取樣錯誤，請在規則中包含 `segment_execution_time > 10` 。 有效值為 0–86,388。
Spectrum 掃描的資料列數	spectrum_scan_row_count	Amazon Redshift Spectrum 查詢在 Amazon S3 中掃描的資料列數。 有效值為 0–999,999,999,999,999。
Spectrum 掃描大小	spectrum_scan_size_mb	Amazon Redshift Spectrum 查詢在 Amazon S3 中掃描的資料大小 (以 MB 為單位)。 有效值為 0–999,999,999,999,999。

指標	名稱	描述
查詢優先順序	query_priority	查詢的優先順序。 有效值 為HIGHEST、HIGH、NORMAL、LOW、LOWEST。 使用大於 (>) 和小於 (<) 運算子比較 query_priority 時，HIGHEST 大於 HIGH、HIGH 大於 NORMAL，以此類推。

> **Note**
>
> - query_queue_time 述詞不支援跳轉動作。也就是說，會忽略定義為在符合 query_queue_time 述詞時進行跳轉的規則。
> - 區段執行時間太短可能會導致某些指標取樣錯誤，例如 io_skew 和 query_cpu_usage_percent。為了避免或減少取樣錯誤，請在規則中包含區段執行時間。segment_execution_time > 10 是很理想的起點。

SVL_QUERY_METRICS 檢視顯示已完成之查詢的指標。SVL_QUERY_METRICS_SUMMARY 檢視顯示已完成之查詢的指標最大值。請使用這些檢視中的值來協助決定臨界值，以定義查詢監控規則。

Amazon Redshift Serverless 的查詢監控指標

下表描述 Amazon Redshift Serverless 查詢監控規則中使用的指標。

指標	名稱	描述
查詢 CPU 時間	max_query_cpu_time	查詢所用的 CPU 時間 (秒)。CPU time 不同於 Query execution time。 有效值為 0–999,999。
已讀取的區塊	max_query_blocks_read	查詢所讀取的 1 MB 資料區塊數。 有效值為 0–1,048,575。

指標	名稱	描述
掃描的資料列數	max_scan_row_count	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。 有效值為 0–999,999,999,999,999。
查詢執行時間	max_query_execution_time	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。如果查詢超過設定的執行時間，Amazon Redshift Serverless 會停止查詢。 有效值為 0–86,399。
查詢佇列時間	max_query_queue_time	在佇列中等待的時間 (秒)。 有效值為 0–86,399。
CPU 用量	max_query_cpu_usage_percent	查詢所用的 CPU 容量百分比。 有效值為 0–6,399。
記憶體到磁碟	max_query_temp_blocks_to_disk	用來寫入中間結果的暫時磁碟空間，以 1 MB 區塊為單位。 有效值為 0–319,815,679。
已聯結的資料列	max_join_row_count	聯結步驟中處理的資料列數。 有效值為 0–999,999,999,999,999。
巢狀迴圈聯結資料列數	max_nested_loop_join_row_count	巢狀迴圈聯結中的數字或資料列。 有效值為 0–999,999,999,999,999。

> **Note**
>
> - max_query_queue_time 述詞不支援跳轉動作。也就是說，會忽略定義為在符合 max_query_queue_time 述詞時進行跳轉的規則。
> - 區段執行時間太短可能會導致某些指標取樣錯誤，例如 max_io_skew 和 max_query_cpu_usage_percent。

查詢監控規則範本

當您使用 Amazon Redshift 主控台新增規則時，您可以選擇從預先定義的範本建立規則。Amazon Redshift 會建立包含一組述詞的新規則，並以預設值填入述詞。預設動作為記錄。您可以修改述詞和動作，以符合您的使用案例。

下表列出可用的範本。

範本名稱	述詞	描述
巢狀迴路聯結	nested_loop_join_row_count > 100	巢狀迴圈聯結可能表示聯結述詞不完整，通常會產生非常大的傳回集 (笛卡兒乘積)。請使用較小的資料列數，以儘早發現潛在的失控查詢。
查詢傳回的大量的資料列數	return_row_count > 1000000	如果您將佇列專用於簡單、短期執行的查詢，您可以包含規則來找出傳回較多資料列數的查詢。範本以 1 百萬個資料列為預設值。對於某些系統，您可能認為 1 百萬個資料列太多，或在較大的系統中，十億或更多個資料列可能太多。
一同加入大量的資料列數	join_row_count > 1000000000	涉及非常多資料列的聯結步驟可能表示需要更嚴格的篩選條件。範本以十億個資料列為預設值。對於用於快速、簡單查詢的臨機操作 (一次性) 佇列，您可以使用較小的數目。
寫入中繼結果時出現高磁碟用量	query_temp_blocks_to_disk > 100000	當目前執行的查詢使用超過可用的系統 RAM 時，查詢執行引擎會將中間結果寫入磁碟 (記憶體外溢)。這種情況通常是惡意查詢所引起，這通常也是使用最多磁碟空間的查詢。可接受的磁

範本名稱	述詞	描述
		碟用量臨界值根據叢集節點類型和節點數而有所不同。範本以 100,000 個區塊或 100 GB 為預設值。對於小型叢集，您可以使用較小的數目。
查詢執行時間長且 I/O 高度扭曲	`segment_execution_time > 120` 和 `io_skew > 1.30`	當一個節點配量的 I/O 速率比其他配量高出許多時，就會發生 I/O 扭曲。以扭曲 1.30 (平均值的 1.3 倍) 為起點應該太高。I/O 扭曲偏高不一定是問題，但加上執行查詢時間太長時，就可能表示分佈樣式或排序索引鍵有問題。

查詢監控規則的系統資料表和檢視

當符合規則的所有述詞時，WLM 會將一列寫入 STL_WLM_RULE_ACTION 系統資料表。這一列包含觸發規則的查詢和所產生動作的詳細資訊。

此外，Amazon Redshift 會在下列系統資料表和檢視中記錄查詢指標。

- STV_QUERY_METRICS 資料表會顯示目前執行之查詢的指標。
- STL_QUERY_METRICS 資料表會記錄以完成之查詢的指標。
- SVL_QUERY_METRICS 檢視顯示已完成之查詢的指標。
- SVL_QUERY_METRICS_SUMMARY 檢視顯示已完成之查詢的指標最大值。

WLM 系統資料表和檢視

WLM 會根據內部定義的 WLM 服務類別來設定查詢佇列。Amazon Redshift 會根據這些服務類別以及 WLM 組態中定義的佇列，建立多個內部佇列。在系統資料表中，佇列和服務類別這兩個詞通常互通使用。超級使用者佇列使用服務類別 5。使用者定義的查詢使用服務類別 6 以上。

您可以使用 WLM 專屬系統資料表來檢視查詢、佇列和服務類別的狀態。查詢下列系統資料表來執行下列動作：

- 檢視工作負載管理員正在追蹤哪些查詢及已配置什麼資源。
- 查看已將查詢指派給哪一個佇列。
- 檢視工作負載管理員目前追蹤之查詢的狀態。

資料表名稱	描述
STL_WLM_ERROR	包含 WLM 相關錯誤事件的日誌。
STL_WLM_QUERY	列出 WLM 正在追蹤的查詢。
STV_WLM_CLASSIFICATION_CONFIG	顯示 WLM 目前的分類規則。
STV_WLM_QUERY_QUEUE_STATE	記錄查詢佇列的目前狀態。
STV_WLM_QUERY_STATE	提供 WLM 正在追蹤之查詢的目前狀態快照。
STV_WLM_QUERY_TASK_STATE	包含查詢任務的目前狀態。
STV_WLM_SERVICE_CLASS_CONFIG	記錄 WLM 的服務類別組態。
STV_WLM_SERVICE_CLASS_STATE	包含服務類別的目前狀態。
STL_WLM_RULE_ACTION	記錄 WLM 查詢監控規則所產生之動作的詳細資訊，而此規則與使用者定義的查詢相關聯。
STV_WLM_QMR_CONFIG	記錄 WLM 查詢監控規則 (QMR) 的組態設定。

您可以在系統資料表中使用任務 ID 來追蹤查詢。以下範例示範如何取得最近提交之使用者查詢的任務 ID：

```
select task from stl_wlm_query where exec_start_time =(select max(exec_start_time) from
 stl_wlm_query);

task
------
137
(1 row)
```

下列範例顯示目前正在執行或在各種服務類別 (佇列) 中等待的查詢。此查詢很適用於追蹤 Amazon Redshift 的整體並行工作負載：

```
select * from stv_wlm_query_state order by query;


xid |task|query|service_| wlm_start_  |  state  |queue_ | exec_
    |    |     |class   | time        |         |time   | time
----+----+-----+--------+-------------+---------+-------+--------
2645| 84 | 98  | 3      | 2010-10-... |Returning|   0   | 3438369
2650| 85 | 100 | 3      | 2010-10-... |Waiting  |   0   | 1645879
2660| 87 | 101 | 2      | 2010-10-... |Executing|   0   | 916046
2661| 88 | 102 | 1      | 2010-10-... |Executing|   0   | 13291
(4 rows)
```

WLM 服務類別 ID

下表列出指派給服務類別的 ID。

ID	服務類別
1–4	保留以供系統使用。
5	供超級使用者佇列使用。
6–13	供 WLM 組態中定義的手動 WLM 佇列使用。
14	供短期查詢加速使用。
15	保留給 Amazon Redshift 執行的維護活動。
100–107	當 auto_wlm 為 true 時供自動 WLM 佇列使用。

管理資料庫安全

主題

您可以控制哪些使用者可以存取哪些資料庫物件來管理資料庫安全性。

能否存取資料庫物件取決於您授予使用者或群組的許可。下列準則彙總資料庫安全性的運作方式：

- 依預設，只對物件擁有者授予許可。
- Amazon Redshift 資料庫使用者是可以連接至資料庫的具名使用者。使用者會以兩種方式獲得許可：直接明確地或隱含地將這些許可指派給帳戶、成為已獲得許可的群組成員。
- 群組是可以集體獲指派許可，以簡化安全維護的使用者集合。
- 結構描述是資料庫資料表和其他資料庫物件的集合。結構描述與檔案系統目錄相似，但結構描述無法巢狀化。使用者可獲得存取單一結構描述或多個結構描述的權限。

此外，Amazon Redshift 還採用下列功能，讓您更好地控制哪些使用者可以存取哪些資料庫物件：

- 角色型存取控制 (RBAC) 可讓您將許可指派給角色，然後再套用至使用者，讓您控制大型使用者群組的許可。與群組不同，角色可以繼承其他角色的許可。

  資料列層級安全 (RLS) 可讓您定義政策，限制所選擇資料列的存取權，然後將這些政策套用至使用者或群組。

動態資料遮罩 (DDM) 可在查詢執行期轉換資料，進一步保護您的資料，讓您可以允許使用者存取資料，但不會暴露敏感詳細資料。

如需安全性實作的範例，請參閱控制使用者和群組存取的範例。

如需保護資料的相關資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的安全性。

Amazon Redshift 安全性概觀

Amazon Redshift 資料庫安全性不同於其他類型的 Amazon Redshift 安全性。除了本節描述的資料庫安全性外，Amazon Redshift 還提供這些功能來管理安全性：

- 登入登入資料 — 對 Amazon Redshift AWS 管理主控台的存取權由您的 AWS 帳戶許可控制。如需詳細資訊，請參閱登入資料。
- 存取管理 — 若要控制對特定 Amazon Redshift 資源的存取，請定義 AWS Identity and Access Management (IAM) 帳戶。(如需詳細資訊，請參閱控制 Amazon Redshift 資源的存取)。
- 叢集安全群組 — 若要授予其他使用者 Amazon Redshift 叢集的傳入存取，您可以定義叢集安全群組，並將它與叢集建立關聯。如需詳細資訊，請參閱 Amazon Redshift 叢集安全群組。
- VPC — 若要使用虛擬聯網環境來保護叢集的存取，您可以在 Amazon 虛擬私有雲端 (VPC) 啟動叢集。如需詳細資訊，請參閱管理虛擬私有雲端 (VPC)中的叢集。
- 叢集加密 — 若要加密您的所有使用者建立的資料表中的資料，您可以在啟動叢集時開啟叢集加密。如需詳細資訊，請參閱 Amazon Redshift 叢集。
- SSL 連線 — 若要加密您的 SQL 用戶端與您的叢集之間的連線，您可以使用 Secure Sockets Layer (SSL) 加密。如需詳細資訊，請參閱使用 SSL 連接至您的叢集。
- 載入資料加密 — 若要在您將資料表載入資料檔案上傳至 Amazon S3 時加密這些檔案，您可以使用伺服器端加密或用戶端加密。當您從伺服器端加密的資料載入時，Amazon S3 會以透明的方式處理解密。當您從用戶端加密的資料載入時，Amazon Redshift COPY 命令會在載入資料表時解密資料。如需詳細資訊，請參閱 將加密資料上傳到 Amazon S3。
- 傳輸中的資料 — 為了保護您在 AWS 雲端內傳輸中的資料，Amazon Redshift 使用硬體加速 SSL 與 Amazon S3 或 Amazon DynamoDB 通訊，以進行複製、卸載、備份和還原作業。
- 資料欄層級存取控制 — 若要讓資料的資料欄層級在 Amazon Redshift 中存取控制，請使用資料欄層級授權及撤銷陳述式，而不需實作檢視型存取控制或使用其他系統。

- 資料列層級安全控制 — 若要具有 Amazon Redshift 中資料的資料列層級安全控制，請建立政策並將政策附加至角色或使用者，限制他們存取政策中定義的資料列。

預設的資料庫使用者許可

當您建立資料庫物件時，您是其擁有者。依預設，只有超級使用者或物件擁有者才能查詢物件、修改物件，或授予物件許可。對於要使用物件的使用者，您必須將必要的許可授予使用者或包含使用者的群組。資料庫超級使用者具有與資料庫擁有者相同的許可。

Amazon Redshift 支援下列許可：
SELECT、INSERT、UPDATE、DELETE、REFERENCES、CREATE、TEMPORARY 和 USAGE。不同的許可與不同的物件類型相關聯。如需有關 Amazon Redshift 支援的資料庫物件許可資訊，請參閱 GRANT 命令。

只有擁有者才有修改或銷毀物件的許可。

根據預設，所有使用者對資料庫的 PUBLIC 結構描述都具有 CREATE 和 USAGE 許可。若要不允許使用者在資料庫的 PUBLIC 結構描述內建立物件，請使用 REVOKE 命令來移除該許可。

若要撤銷先前授予的許可，請使用 REVOKE 命令。物件擁有者的許可 (例如 DROP、GRANT 和 REVOKE 許可) 是隱含的，而且無法授予或撤銷。物件擁有者可以撤銷自己的普通許可，例如，使自己以及其他人只能讀取資料表。無論 GRANT 和 REVOKE 命令，超級使用者都能保留所有許可。

超級使用者

資料庫超級使用者具有與所有資料庫之資料庫擁有者相同的許可。

admin 使用者 (即您啟動叢集時所建立的使用者) 是超級使用者。

您必須是超級使用者才能建立超級使用者。

Amazon Redshift 系統資料表和系統檢視分為只有超級使用者才能看見或所有使用者都能看見。只有超級使用者才能查詢指定為「超級使用者才能看見」的系統資料表和系統檢視。如需相關資訊，請參閱系統資料表和檢視。

超級使用者可以檢視所有目錄資料表。如需相關資訊，請參閱系統目錄資料表。

資料庫超級使用者會略過所有許可檢查。無論 GRANT 和 REVOKE 命令，超級使用者都能保留所有許可。使用超級使用者角色時必須謹慎。建議您以不是超級使用者的角色執行大部分工作。您可以使用更嚴格的許可建立管理員角色。如需有關建立角色的詳細資訊，請參閱 角色型存取控制 (RBAC)

若要建立新的資料庫超級使用者，請以超級使用者身分登入資料庫，然後發出 CREATE USER 命令，或搭配 CREATEUSER 許可的 ALTER USER 命令。

```
CREATE USER adminuser CREATEUSER PASSWORD '1234Admin';
ALTER USER adminuser CREATEUSER;
```

若要建立、修改或捨棄超級使用者，請使用相同的命令來管理使用者。如需詳細資訊，請參閱 建立、更改和刪除使用者。

使用者

您可以使用 Amazon Redshift SQL 命令 CREATE USER 和 ALTER USER 來建立和管理資料庫使用者。或是，您可以使用自訂的 Amazon Redshift JDBC 或 ODBC 驅動程式設定您的 SQL 用戶端。這些驅動程式會管理隨著資料庫登入的一部分建立使用者資料庫和暫時密碼的程序。

驅動程式會根據 AWS Identity and Access Management (IAM) 身份驗證資料庫使用者。如果您已經在以外管理使用者身分 AWS，則可以使用符合 SAML 2.0 標準的身分識別提供者 (IdP) 來管理對 Amazon Redshift 資源的存取。您可以使用 IAM 角色來設定 IdP，並 AWS 允許聯合身分使用者產生臨時資料庫登入資料並登入 Amazon Redshift 資料庫。如需詳細資訊，請參閱使用 IAM 身分驗證產生資料庫使用者登入資料。

Amazon Redshift 使用者帳戶只能由資料庫超級使用者建立和捨棄。使用者在登入 Amazon Redshift 時會進行身份驗證。他們可以擁有資料庫和資料庫物件 (例如資料表)。他們也可以將這些物件的許可授予使用者、群組和結構描述，以控制誰有權存取哪個物件。具有 CREATE DATABASE 權限的使用者可以建立資料庫，並授予那些資料庫的許可。超級使用者具有所有資料庫的資料庫所有權。

建立、更改和刪除使用者

資料庫使用者適用於資料倉儲叢集 (而且不是根據個別資料庫)。

- 若要建立使用者，請使用 CREATE USER 命令。
- 若要建立超級使用者，請使用 CREATE USER 命令與 CREATEUSER 選項搭配。
- 若要移除現有的使用者，請使用 DROP USER 命令。
- 若要變更使用者 (例如變更密碼)，請使用 ALTER USER 命令。
- 若要檢視使用者清單，請查詢 PG_USER 目錄資料表。

```
select * from pg_user;
```

```
  usename   | usesysid | usecreatedb | usesuper | usecatupd |  passwd  | valuntil |
 useconfig
------------+----------+-------------+----------+-----------+----------+----------
+-----------
 rdsdb      |        1 | t           | t        | t         | ******** |          |
 masteruser |      100 | t           | t        | f         | ******** |          |
 dwuser     |      101 | f           | f        | f         | ******** |          |
 simpleuser |      102 | f           | f        | f         | ******** |          |
 poweruser  |      103 | f           | t        | f         | ******** |          |
 dbuser     |      104 | t           | f        | f         | ******** |          |
(6 rows)
```

群組

群組是全部被授予與群組相關聯之任何許可的使用者集合。您可以使用群組來指派許可。例如，您可以為銷售、管理和支援建立不同的群組，並給與每個群組中的使用者適當權限，讓他們可在工作時存取所需的資料。您可以授予或撤銷群組層級的許可，而且那些變更將適用於群組的所有成員，但超級使用者除外。

若要檢視所有使用者群組，請查詢 PG_GROUP 系統目錄資料表：

```
select * from pg_group;
```

例如，若要依群組列出所有資料庫使用者，請執行下列 SQL。

```
SELECT u.usesysid
,g.groname
,u.usename
FROM pg_user u
LEFT JOIN pg_group g ON u.usesysid = ANY (g.grolist)
```

建立、更改和刪除群組

只有超級使用者才能建立、更改或捨棄群組。

您可以執行下列動作：

- 若要建立群組，請使用 CREATE GROUP 命令。
- 若要將使用者新增至現有群組或從中移除使用者，請使用 ALTER GROUP 命令。

- 若要刪除群組，請使用 DROP GROUP 命令。此命令只會捨棄群組，不會捨棄其成員使用者。

控制使用者和群組存取的範例

此範例會建立使用者群組和使用者，然後將連接至 Web 應用程式用戶端之 Amazon Redshift 資料庫的各種許可授予他們。此範例採用三個使用者群組：Web 應用程式的一般使用者、Web 應用程式的進階使用者，以及 Web 開發人員。

1. 建立將在其中指派使用者的群組。下列一組命令會建立三個不同的使用者群組：

```
create group webappusers;

create group webpowerusers;

create group webdevusers;
```

2. 建立數個具有不同許可的資料庫使用者，並將他們新增至群組。
   a. 建立兩個使用者，並將他們新增至 WEBAPPUSERS 群組：

```
create user webappuser1 password 'webAppuser1pass'
in group webappusers;

create user webappuser2 password 'webAppuser2pass'
in group webappusers;
```

   b. 建立 Web 開發人員使用者，並將其新增至 WEBDEVUSERS 群組：

```
create user webdevuser1 password 'webDevuser2pass'
in group webdevusers;
```

   c. 建立一個超級使用者。此使用者將具有建立其他使用者的管理權限：

```
create user webappadmin  password 'webAppadminpass1'
createuser;
```

3. 建立要與 Web 應用程式所使用之資料庫資料表相關聯的結構描述，並將各種使用者群組存取授予這個結構描述：
   a. 建立 WEBAPP 結構描述：

```
create schema webapp;
```

b. 將 USAGE 許可授予 WEBAPPUSERS 群組：

```
grant usage on schema webapp to group webappusers;
```

c. 將 USAGE 許可授予 WEBPOWERUSERS 群組：

```
grant usage on schema webapp to group webpowerusers;
```

d. 將 ALL 許可授予 WEBDEVUSERS 群組：

```
grant all on schema webapp to group webdevusers;
```

基本使用者和群組現在已完成設定。您現在可以修改使用者和群組。

4. 例如，下列命令會更改 WEBAPPUSER1 的 search_path 參數。

```
alter user webappuser1 set search_path to webapp, public;
```

在未指定結構描述的情況下，當簡單名稱參考物件時，SEARCH_PATH 會指定資料庫物件 (例如資料表和函數) 的結構描述搜尋順序。

5. 您也可以在建立群組之後將使用者新增至群組，例如將 WEBAPPUSER2 新增至 WEBPOWERUSERS 群組：

```
alter group webpowerusers add user webappuser2;
```

結構描述

資料庫包含一或多個具名結構描述。資料庫中的每個結構描述都包含資料表和其他類型的具名物件。依預設，資料庫具有名為 PUBLIC 的單一結構描述。您可以使用結構描述，在常見名稱下將資料庫物件分組。結構描述與檔案系統目錄相似，但結構描述無法巢狀化。

相同的資料庫物件名稱可在相同資料庫的不同結構描述中使用，而不會發生衝突。例如，MY_SCHEMA 和 YOUR_SCHEMA 都可以包含名為 MYTABLE 的資料表。具有必要許可的使用者可以跨資料庫中的多個結構描述存取物件。

依預設，物件是在資料庫搜尋路徑中的第一個結構描述內建立。如需相關資訊，請參閱本節後續的搜尋路徑。

結構描述可以協助在多使用者環境中採用下列方式解決組織和並行問題：

- 讓多個開發人員可在同一個資料庫中工作，而彼此不會互相干擾。
- 將資料庫物件組織成邏輯群組，讓它們更便於管理。
- 讓應用程式能夠將其物件放入個別結構描述中，以便其名稱將不會與其他應用程式使用的物件名稱發生衝突。

建立、更改和刪除結構描述

任何使用者都可以建立結構描述，以及更改或捨棄他們擁有的結構描述。

您可以執行下列動作：

- 若要建立結構描述，請使用 CREATE SCHEMA 命令。
- 若要變更結構描述的擁有者，請使用 ALTER SCHEMA 命令。
- 若要刪除結構描述及其物件，請使用 DROP SCHEMA命令。
- 若要在結構描述內建立資料表，請建立格式為 schema_name.table_name 的資料表。

若要檢視所有結構描述的清單，請查詢 PG_NAMESPACE 系統目錄資料表：

```
select * from pg_namespace;
```

若要檢視屬於結構描述的資料表清單，請查詢 PG_TABLE_DEF 系統目錄資料表。例如，下列查詢會傳回 PG_CATALOG 結構描述的資料表清單。

```
select distinct(tablename) from pg_table_def
where schemaname = 'pg_catalog';
```

搜尋路徑

搜尋路徑是在 search_path 參數中搭配逗號分隔的結構描述名稱清單定義的。當以未包括結構描述限定詞的簡單名稱參考物件 (如資料表或函數) 時，此搜尋路徑會指定搜尋結構描述的順序。

如果在未指定目標結構描述的情況下建立物件，則物件會新增至搜尋路徑中列出的第一個結構描述。當名稱相同的物件存在於不同結構描述時，未指定結構描述的物件名稱將參考搜尋路徑中的第一個結構描述，其中包含具有該名稱的物件。

若要變更目前工作階段的預設結構描述，請使用 SET 命令。

如需詳細資訊，請參閱《組態參考》中的 search_path 說明。

結構描述型許可

結構描述型許可是由結構描述的擁有者決定：

- 根據預設，所有使用者對資料庫的 PUBLIC 結構描述都具有 CREATE 和 USAGE 許可。若要不允許使用者在資料庫的 PUBLIC 結構描述內建立物件，請使用 REVOKE 命令來移除該許可。
- 除非物件擁有者將 USAGE 許可授予他們，否則使用者無法存取結構描述中他們未擁有的任何物件。
- 如果已將另一個使用者所建立之結構描述的 CREATE 許可授予使用者，則那些使用者便可在該結構描述中建立物件。

角色型存取控制 (RBAC)

透過使用角色型存取控制 (RBAC) 在 Amazon Redshift 中管理資料庫許可，您可以簡化 Amazon Redshift 中的安全許可管理。您可以控制使用者在廣泛或精細層級上執行的動作，以確保敏感資料的存取安全。您也可以控制使用者存取通常僅限超級使用者存取的工作。透過將不同許可指派給不同角色，並將其指派給不同的使用者，您可以更精細地控制使用者存取權。

獲派角色的使用者只能執行其授權指派角色所指定的工作。例如，具有指派角色且具有 CREATE TABLE 和 DROP TABLE 許可的使用者只有執行這些工作的授權。您可以透過授予不同層級的安全許可給不同的使用者，以控制使用者存取權，讓他們能存取工作所需的資料。

RBAC 會根據使用者的角色需求，將最低許可政策套用至使用者，而不論所涉及的物件類型為何。授予和撤銷許可是在角色層級上執行的，而不需要更新個別資料庫物件的許可。

透過 RBAC，您可以建立具有許可的角色，使其可執行過去需要超級使用者許可的命令。使用者只要獲得包含這些許可的角色授權，就可以執行這些命令。同樣地，您也可以建立角色來限制對某些命令的存取，並將角色指派給已獲得該角色授權的超級使用者或使用者。

若要了解 Amazon Redshift RBAC 的運作方式，請觀看以下影片：簡介 Amazon Redshift 中的角色型存取控制 (RBAC)。

角色階層

角色是您可以指派給使用者或其他角色的許可集合。您可以將系統或資料庫許可指派給角色。使用者會繼承指派角色的許可。

在 RBAC 中，使用者可以擁有巢狀角色。您可以將角色授予使用者和角色。將角色授予使用者時，您可以使用此角色包含的所有權限來授權使用者。將角色 r1 授予使用者時，您可以使用 r1 的許可來授權使用者。使用者現在擁有 r1 的許可，以及他們已有的任何現有許可。

將角色 (r1) 授予另一個角色 (r2) 時，您可以使用 r1 的所有許可來授權 r2。此外，將 r2 授予另一個角色 (r3) 時，r3 的許可會結合 r1 和 r2 的許可。角色階層會讓 r2 從 r1 繼承許可。Amazon Redshift 會透過每個角色授權來傳播許可。將 r1 授予 r2，然後再將 r2 授予 r3，即可將這三個角色的所有許可授權給 r3。因此，透過將 r3 授予使用者，使用者就會擁有這三個角色的所有許可。

Amazon Redshift 不允許建立角色授權循環。當巢狀角色被指派回角色階層中較早的角色 (例如 r3 被指派回 r1) 時，就會發生角色授權循環。如需建立角色及管理角色指派的相關資訊，請參閱 管理 RBAC 中的角色。

角色指派

超級使用者和擁有 CREATE ROLE 許可的一般使用者可以使用 CREATE ROLE 陳述式來建立角色。超級使用者和角色管理員可以使用 GRANT ROLE 陳述式將角色授予其他人。他們可以使用 REVOKE ROLE 陳述式來撤銷其他人的角色，以及使用 DROP ROLE 陳述式來捨棄角色。角色管理員包括角色擁有者和獲派角色具有 ADMIN OPTION 許可的使用者。

只有超級使用者或角色管理員可以授予和撤銷角色。您可以對一個或多個角色或使用者授予或撤銷一個或多個角色。在 GRANT ROLE 陳述式中使用 WITH ADMIN OPTION 選項會為所有承授者提供所有授予角色的管理選項。

Amazon Redshift 支援不同的角色指派組合，例如授予多個角色或擁有多個承授者。WITH ADMIN OPTION 僅適用於使用者，而不適用於角色。同樣地，使用 REVOKE ROLE 陳述式中的 WITH ADMIN OPTION 選項，可移除承授者的角色和管理授權。與 ADMIN OPTION 搭配使用時，只能撤銷角色的管理授權。

下列範例會從 `user2` 中撤銷 `sample_role2` 角色的管理授權。

```
REVOKE ADMIN OPTION FOR sample_role2 FROM user2;
```

如需建立角色及管理角色指派的相關資訊，請參閱 管理 RBAC 中的角色。

Amazon Redshift 系統定義角色

Amazon Redshift 會提供使用特定許可定義的系統定義角色。系統專屬角色會以 `sys:` 字首開頭。只有具有適當存取權的使用者才能修改系統定義角色或建立自訂的系統定義角色。您無法針對自訂系統定義角色使用 `sys:` 字首。

下表摘要說明角色及其許可。

角色名稱	描述		
sys:monitor	此角色具有存取目錄或系統資料表的許可。		
sys:operator	此角色具有存取目錄或系統資料表、分析、清除或取消查詢的許可。		
sys:dba	此角色具有建立結構描述、建立資料表、捨棄結構描述、捨棄資料表和截斷資料表的許可。其有權建立或取代預存程序、捨棄程序、建立或替代函數、建立或取代外部函數、建立檢視和捨棄檢視。此外，此角色會繼承 sys:operator 角色的所有許可。		
sys:superuser	此角色具有 RBAC 的系統許可 中定義的所有受支援的系統權限。		
sys:secadmin	• 此角色有權建立使用者、修改使用者、捨棄使用者、建立角色、捨棄角色和授予角色許可。 • 此角色有權在關係上開啟或關閉 RLS，以及管理 RLS 和 DDM 政策 (CREATE、DROP、ATTACH、DETACH 和 ALTER)。此外，請注意，依預設會授予此角色 EXPLAIN RLS、IGNORE RLS 和 EXPLAIN MASKING 權限。 • 只有將許可明確授予該角色時，此角色才能存取使用者資料表。		

用於資料共用的系統定義角色和使用者

Amazon Redshift 會建立角色和使用者供內部使用，這些角色和使用者對應於資料存取者和資料清單取用者。每個內部角色名稱和使用者名稱都有保留的命名空間前置詞ds:。它們具有以下格式：

名稱	描述		
ds:*sharei*	與資料相對應的系統角色。		
ds:*sharei* _*consume:*	與資料相對應的使用者的系統使用者。		

系統會為每個資料清單建立資料共用角色。它擁有目前授予資料命令的所有權限。系統會為每個資料存取者建立一個資料共用使用者。它被授與單一資料共用角色的權限。添加到多個數據庫的消費者將為每個數據保護創建一個數據共享用戶。

這些使用者和角色需要資料共用才能正常運作。無法修改或刪除它們，也無法存取或用於客戶執行的任何工作。您可以放心地忽略它們。如需有關資料共用的詳細資訊，請參閱 Amazon Redshift 中的跨叢集共用資料。

> **Note**
>
> 您無法使用ds:前置詞來建立使用者定義的角色或使用者。

RBAC 的系統許可

下列是您可以對角色授予或撤銷的系統許可清單。

Comman	您必須具有下列其中一種方式的權限才能執行命令		
CREATE ROLE	• 超級使用者。 • 具有 CREATE ROLE 許可的使用者。		
DROP ROLE	• 超級使用者。		

Comman	您必須具有下列其中一種方式的權限才能執行命令		
	• 角色擁有者，也就是建立角色的使用者，或是已被授予角色且該角色具有 WITH ADMIN OPTION 許可的使用者。		
CREATE USER	• 超級使用者。 • 具有 CREATE USER 許可的使用者。這些使用者無法建立超級使用者。		
DROP USER	• 超級使用者。 • 具有 DROP USER 許可的使用者。		
ALTER USER	• 超級使用者。 • 具有 ALTER USER 許可的使用者。這些使用者無法將使用者變更為超級使用者或將超級使用者變更為使用者。 • 想要變更自己密碼的目前使用者。		
CREATE SCHEMA	• 超級使用者。 • 具有 CREATE SCHEMA 許可的使用者。		
DROP SCHEMA	• 超級使用者。 • 具有 DROP SCHEMA 許可的使用者。 • 結構描述擁有者。		
ALTER DEFAULT PRIVILEGES	• 超級使用者。 • 具有 ALTER DEFAULT PRIVILEGES 許可的使用者。 • 變更自身預設存取許可的使用者 • 為有權存取的結構描述設定許可的使用者。		
CREATE TABLE	• 超級使用者。 • 具有 CREATE TABLE 許可的使用者。 • 在結構描述上具有 CREATE 許可的使用者。		
DROP TABLE	• 超級使用者。 • 具有 DROP TABLE 許可的使用者。 • 在結構描述上具有 USAGE 許可的資料表擁有者。		

Comman	您必須具有下列其中一種方式的權限才能執行命令
ALTER TABLE	• 超級使用者。 • 具有 ALTER TABLE 許可的使用者。 • 在結構描述上具有 USAGE 許可的資料表擁有者。

CREATE OR REPLAC FUNCTIC	• 對於 CREATE FUNCTION：
	  • 超級使用者。
	  • 具有 CREATE OR REPLACE FUNCTION 許可的使用者
	  • 具有語言 USAGE 許可的使用者。
	• 對於 REPLACE FUNCTION：
	  • 超級使用者。
	  • 具有 CREATE OR REPLACE FUNCTION 許可的使用者
	  • 函數擁有者。

CREATE OR REPLAC EXTERN FUNCTIC	• 超級使用者。 • 具有 CREATE OR REPLACE EXTERNAL FUNCTION 許可的使用者。
DROP FUNCTIC	• 超級使用者。 • 具有 DROP FUNCTION 許可的使用者。 • 函數擁有者。

CREATE OR REPLAC PROCED	• 對於 CREATE PROCEDURE：
	  • 超級使用者。
	  • 具有 CREATE OR REPLACE PROCEDURE 許可的使用者。
	  • 具有語言 USAGE 許可的使用者。
	• 對於 REPLACE PROCEDURE：
	  • 超級使用者。
	  • 具有 CREATE OR REPLACE PROCEDURE 許可的使用者。
	  • 程序擁有者。

Comman	您必須具有下列其中一種方式的權限才能執行命令		
DROP PROCED	• 超級使用者。 • 具有 DROP PROCEDURE 許可的使用者。 • 程序擁有者。		

CREATE OR REPLAC VIEW	• 對於 CREATE VIEW：
	  • 超級使用者。
	  • 具有 CREATE OR REPLACE VIEW 許可的使用者
	  • 在結構描述上具有 CREATE 許可的使用者。
	• 對於 REPLACE VIEW：
	  • 超級使用者。
	  • 具有 CREATE OR REPLACE VIEW 許可的使用者
	  • 檢視擁有者。		

DROP VIEW	• 超級使用者。 • 具有 DROP VIEW 許可的使用者。 • 檢視擁有者。		
CREATE MODEL	• 超級使用者。 • 具有 CREATE MODEL 系統許可的使用者，可讀取 CREATE MODEL 的關聯性。 • 具有 CREATE MODEL 許可的使用者。		
DROP MODEL	• 超級使用者。 • 具有 DROP MODEL 許可的使用者。 • 模型擁有者。 • 結構描述擁有者。		
CREATE DATASH	• 超級使用者。 • 具有 CREATE DATASHARE 許可的使用者。 • 資料庫擁有者。		

Comman	您必須具有下列其中一種方式的權限才能執行命令		
ALTER DATASH	• 超級使用者。 • 具有 ALTER DATASHARE 許可的使用者。 • 在資料共用上具有 ALTER 或 ALL 許可的使用者。 • 若要將特定物件新增至資料共用，這些使用者必須要有物件的許可。使用者必須是物件的擁有者，或具有物件的 SELECT、USAGE 或 ALL 許可。		
DROP DATASH	• 超級使用者。 • 具有 DROP DATASHARE 許可的使用者。 • 資料庫擁有者。		
CREATE LIBRARY	• 超級使用者。 • 具有 CREATE LIBRARY 許可或具有指定語言許可的使用者。		
DROP LIBRARY	• 超級使用者。 • 具有 DROP LIBRARY 許可的使用者。 • 程式庫擁有者。		
ANALYZI	• 超級使用者。 • 具有 ANALYZE 許可的使用者。 • 關係的擁有者。 • 共用資料表的資料庫擁有者。		
取消	• 超級使用者取消自己的查詢。 • 超級使用者取消使用者的查詢。 • 具有 CANCEL 許可的使用者取消使用者的查詢。 • 使用者取消自己的查詢。		
TRUNCA TABLE	• 超級使用者。 • 具有 TRUNCATE TABLE 許可的使用者。 • 資料表擁有者。		

Comman	您必須具有下列其中一種方式的權限才能執行命令		
VACUUM	• 超級使用者。 • 具有 VACUUM 許可的使用者。 • 資料表擁有者。 • 共用資料表的資料庫擁有者。		
IGNORE RLS	• 超級使用者。 • `sys:secadmin` 角色中的使用者。		
EXPLAIN RLS	• 超級使用者。 • `sys:secadmin` 角色中的使用者。		
EXPLAIN MASKING	• 超級使用者。 • `sys:secadmin` 角色中的使用者。		

資料庫物件許可

除了系統許可外，Amazon Redshift 還包含可定義存取選項的資料庫物件許可。其中包含的選項包括能夠讀取資料表和檢視中的資料、寫入資料、建立資料表及捨棄資料表。如需詳細資訊，請參閱 GRANT 命令。

透過使用 RBAC，您可以將資料庫物件許可指派給角色，類似於使用系統許可的方式。然後，您可以將角色指派給使用者、授權具有系統許可的使用者，以及使用資料庫許可授權使用者

RBAC 的 ALTER DEFAULT PRIVILEGES

使用 ALTER DEFAULT PRIVILEGES 陳述式來定義日後由指定使用者建立的物件將套用的存取許可預設設定。根據預設，使用者只能變更自己擁有的預設存取許可。使用 RBAC，您可以設定角色的預設存取許可。如需詳細資訊，請參閱 ALTER DEFAULT PRIVILEGES 命令。

RBAC 可讓您將資料庫物件許可指派給角色，類似於系統許可。然後，您可以將角色指派給使用者、使用系統和/或資料庫許可授權使用者。

RBAC 中角色使用的考量

使用 RBAC 角色時，請考慮以下事項：

- Amazon Redshift 不允許循環的角色授權。您不能將 r1 授予 r2，然後又將 r2 授予 r1。
- RBAC 適用於原生 Amazon Redshift 物件和 Amazon Redshift Spectrum 資料表。
- 身為 Amazon Redshift 管理員，您可以將叢集升級至最新的維護修補程式，以開始使用 RBAC。
- 只有超級使用者和擁有 CREATE ROLE 系統許可的使用者才能建立角色。
- 只有超級使用者和角色管理員可以修改和捨棄角色。
- 角色名稱不能與使用者名稱相同。
- 角色名稱不能包含無效字元，例如「:/\n」。
- 角色名稱不能是保留字，例如 PUBLIC。
- 角色名稱不能以預設角色的保留字首開頭 (sys:)。
- 將具有 RESTRICT 參數的角色授予其他角色時，您無法刪除該角色。預設設定為 RESTRICT。當您嘗試捨棄繼承另一個角色的角色時，Amazon Redshift 會擲回錯誤。
- 沒有角色管理員許可的使用者無法予與或撤銷角色。

管理 RBAC 中的角色

欲執行下列動作，請使用下列命令：

- 若要建立角色，請使用 CREATE ROLE 命令。
- 若要重新命名角色或變更角色的擁有者，請使用 ALTER ROLE 命令。
- 若要刪除角色，請使用 DROP ROLE 命令。
- 若要將角色授予使用者，請使用 GRANT 命令。
- 若要撤銷使用者的角色，請使用 REVOKE 命令。
- 若要將系統許可授予角色，請使用 GRANT 命令。
- 若要撤銷角色的系統許可，請使用 REVOKE 命令。

若要檢視叢集或工作群組中的角色清單，請參閱 SVV_ROLES。

教學課程：使用 RBAC 建立角色和查詢

透過 RBAC，您可以建立具有許可的角色，使其可執行過去需要超級使用者許可的命令。使用者只要獲得包含這些許可的角色授權，就可以執行這些命令。

在本教學課程中，您會使用以角色為基礎的存取控制 (RBAC) 來管理您建立的資料庫中的權限。然後，您可以連線到資料庫，並從兩個不同的角色查詢資料庫，以測試 RBAC 的功能。

您建立並用來查詢資料庫的兩個角色是sales_ro和sales_rw。您可以建立sales_ro角色，並以具有該sales_ro角色的使用者身分查詢資料。使用sales_ro者只能使用 SELECT 指令，但無法使用 UPDATE 指令。然後，您可以建立sales_rw角色並以具有該sales_rw角色的使用者身分查詢資料。用sales_rw戶可以使用 SELECT 命令和更新命令。

此外，您可以建立角色來限制對某些指令的存取，並將角色指派給超級使用者或使用者。

工作

- 必要條件
- 步驟 1：建立管理員使用者
- 步驟 2：設定綱要
- 步驟 3：建立唯讀使用者
- 步驟 4：以唯讀使用者身分查詢資料
- 步驟 5：建立讀寫使用者
- 步驟 6：以具有繼承唯讀角色的使用者身分查詢資料
- 步驟 7：將更新和插入權限授與讀寫角色
- 步驟 8：以讀寫使用者身分查詢資料
- 步驟 9：以管理員使用者身分分析資料庫中的資料表並加以真空
- 步驟 10：以讀寫使用者的身分截斷資料表

必要條件

- 建立已載入 TICKIT 範例資料庫的 Amazon Redshift 叢集或無伺服器工作群組。若要建立無伺服器工作群組，請參閱 Amazon Redshift 無伺服器。若要建立叢集，請參閱建立範例 Amazon Redshift 叢集。如需 TICKIT 範例庫的相關資訊，請參閱 範本資料庫。
- 具有超級使用者或角色管理員權限的使用者的存取權。只有超級使用者或角色管理員可以授與或撤銷角色。如需 RBAC 所需權限的詳細資訊，請參閱。RBAC 的系統許可
- 請參閱RBAC 中角色使用的考量。

步驟 1：建立管理員使用者

若要設定此教學課程，請在此步驟中建立資料庫管理員角色，並將其附加至資料庫管理員使用者。您必須將資料庫管理員建立為超級使用者或角色管理員。

在 Amazon Redshift https://docs.aws.amazon.com/redshift/latest/mgmt/query-editor-v2-using.html 中運行所有查詢。

1. 若要建立管理員角色 db_admin，請使用下列範例。

```
CREATE ROLE db_admin;
```

2. 若要建立名為 dbadmin 的資料庫使用者，請使用下列範例。

```
CREATE USER dbadmin PASSWORD 'Test12345';
```

3. 若要將名為 sy: dba 的系統定義角色授與 db_admin 角色，請使用下列範例。當被授與 sy: dba 角色時，dbadmin 使用者就可以建立結構描述和表格。如需詳細資訊，請參閱 Amazon Redshift 系統定義角色。

步驟 2：設定綱要

在此步驟中，您會以資料庫管理員的身分連線到資料庫。然後，您可以建立兩個資料架構，並將資料加入至其中。

1. 使用查詢編輯器 v2 以 dbadmin 使用者身分 Connect 線到 dev 資料庫。如需有關連線至資料庫的詳細資訊，請參閱使用查詢編輯器 v2。
2. 若要建立銷售和行銷資料庫結構描述，請使用下列範例。

```
CREATE SCHEMA sales;
CREATE SCHEMA marketing;
```

3. 若要建立值並將其插入至 sales 架構中的資料表，請使用下列範例。

```
CREATE TABLE sales.cat(
catid smallint,
catgroup varchar(10),
catname varchar(10),
catdesc varchar(50)
);
INSERT INTO sales.cat(SELECT * FROM category);

CREATE TABLE sales.dates(
dateid smallint,
caldate date,
day char(3),
```

```
week smallint,
month char(5),
qtr char(5),
year smallint,
holiday boolean
);
INSERT INTO sales.dates(SELECT * FROM date);

CREATE TABLE sales.events(
eventid integer,
venueid smallint,
catid smallint,
dateid smallint,
eventname varchar(200),
starttime timestamp
);
INSERT INTO sales.events(SELECT * FROM event);

 CREATE TABLE sales.sale(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp
);
INSERT INTO sales.sale(SELECT * FROM sales);
```

4. 若要在行銷結構描述中建立值並將其插入資料表，請使用下列範例。

```
CREATE TABLE marketing.cat(
catid smallint,
catgroup varchar(10),
catname varchar(10),
catdesc varchar(50)
);
INSERT INTO marketing.cat(SELECT * FROM category);

CREATE TABLE marketing.dates(
dateid smallint,
```

```
caldate date,
day char(3),
week smallint,
month char(5),
qtr char(5),
year smallint,
holiday boolean
);
INSERT INTO marketing.dates(SELECT * FROM date);

CREATE TABLE marketing.events(
eventid integer,
venueid smallint,
catid smallint,
dateid smallint,
eventname varchar(200),
starttime timestamp
);
INSERT INTO marketing.events(SELECT * FROM event);

CREATE TABLE marketing.sale(
marketingid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp
);
INSERT INTO marketing.sale(SELECT * FROM marketing);
```

步驟 3：建立唯讀使用者

在此步驟中，您會為唯讀角色建立唯讀角色和 salesAnalyst 使用者。銷售分析師只需要對銷售架構中表格的唯讀存取權，即可完成其指派的任務，以尋找產生最大佣金的事件。

1. 以 dbadmin 使用者的身分 Connect 至資料庫。
2. 若要建立 sales_ro 角色，請使用下列範例。

```
CREATE ROLE sales_ro;
```

3. 若要建立銷售分析員使用者，請使用下列範例。

```
CREATE USER salesanalyst PASSWORD 'Test12345';
```

4. 若要授與 sales_ro 角色用法，並選取對 sales 綱要物件的存取權，請使用下列範例。

```
GRANT USAGE ON SCHEMA sales TO ROLE sales_ro;
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO ROLE sales_ro;
```

5. 若要授與銷售分析員使用者 sales_ro 角色，請使用下列範例。

```
GRANT ROLE sales_ro TO salesanalyst;
```

步驟 4：以唯讀使用者身分查詢資料

在此步驟中，銷售分析員使用者會從銷售架構查詢資料。然後，salesAnalyst 使用者會嘗試更新資料表並讀取行銷結構描述中的資料表。

1. 以銷售分析員使用者身分 Connect 至資料庫。
2. 要查找佣金最高的 10 筆銷售，請使用以下示例。

```
SET SEARCH_PATH TO sales;
SELECT DISTINCT events.dateid, sale.commission, cat.catname
FROM sale, events, dates, cat
WHERE events.dateid=dates.dateid AND events.dateid=sale.dateid AND events.catid =
 cat.catid
ORDER BY 2 DESC LIMIT 10;

+--------+------------+----------+
| dateid | commission | catname  |
+--------+------------+----------+
|   1880 |     1893.6 | Pop      |
|   1880 |     1893.6 | Opera    |
|   1880 |     1893.6 | Plays    |
|   1880 |     1893.6 | Musicals |
|   1861 |       1500 | Plays    |
|   2003 |       1500 | Pop      |
|   1861 |       1500 | Opera    |
```

```
|   2003 |       1500 | Plays    |
|   1861 |       1500 | Musicals |
|   1861 |       1500 | Pop      |
+--------+------------+----------+
```

3. 若要從 sales 結構描述的事件表中選取 10 個事件，請使用下列範例。

```
SELECT * FROM sales.events LIMIT 10;

+---------+---------+-------+--------+--------------------+---------------------+
| eventid | venueid | catid | dateid |     eventname      |      starttime      |
+---------+---------+-------+--------+--------------------+---------------------+
|    4836 |      73 |     9 |   1871 | Soulfest           | 2008-02-14 19:30:00 |
|    5739 |      41 |     9 |   1871 | Fab Faux           | 2008-02-14 19:30:00 |
|     627 |     229 |     6 |   1872 | High Society       | 2008-02-15 14:00:00 |
|    2563 |     246 |     7 |   1872 | Hamlet             | 2008-02-15 20:00:00 |
|    7703 |      78 |     9 |   1872 | Feist              | 2008-02-15 14:00:00 |
|    7903 |      90 |     9 |   1872 | Little Big Town    | 2008-02-15 19:30:00 |
|    7925 |     101 |     9 |   1872 | Spoon              | 2008-02-15 19:00:00 |
|    8113 |      17 |     9 |   1872 | Santana            | 2008-02-15 15:00:00 |
|     463 |     303 |     8 |   1873 | Tristan und Isolde | 2008-02-16 19:00:00 |
|     613 |     236 |     6 |   1873 | Pal Joey           | 2008-02-16 15:00:00 |
+---------+---------+-------+--------+--------------------+---------------------+
```

4. 若要嘗試更新 eventid 1 的事件名稱，請執行下列範例。此範例會導致權限遭拒錯誤，因為 salesAnalyst 使用者只有銷售結構描述中事件資料表的 SELECT 權限。若要更新事件資料表，您必須將 sales_ro 角色權限授與更新。如需有關授與更新資料表之權限的詳細資訊，請參閱的 UPDATE 參數GRANT。如需 UPDATE 指令的更多資訊，請參閱〈〉UPDATE。

```
UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;

ERROR: permission denied for relation events
```

5. 若要嘗試從行銷結構描述的事件表中選取所有項目，請使用下列範例。此範例會導致權限遭拒錯誤，因為 salesAnalyst 使用者只有銷售結構描述中事件資料表的 SELECT 權限。若要從行銷結構描述中的事件表中選取資料，您必須授與 sales_ro 角色 SELECT 權限，對行銷結構描述中的事件表格。

```
SELECT * FROM marketing.events;
```

```
ERROR: permission denied for schema marketing
```

步驟 5：建立讀寫使用者

在此步驟中，負責建置擷取、轉換和載入 (ETL) 管道以便在 sales Schema 中進行資料處理的銷售工程師將獲得唯讀存取權限，但稍後會獲得讀取和寫入存取權，以執行其工作。

1. 以 dbadmin 使用者的身分 Connect 至資料庫。
2. 若要在銷售結構描述中建立 sales_rw 角色，請使用下列範例。

```
CREATE ROLE sales_rw;
```

3. 若要建立銷售工程師使用者，請使用下列範例。

```
CREATE USER salesengineer PASSWORD 'Test12345';
```

4. 若要授與 sales_rw 角色用法，並透過指派 sales_ro 角色來選取 sales 綱要物件的存取權，請使用下列範例。如需角色如何在 Amazon Redshift 中繼承權限的詳細資訊，請參閱角色階層。

```
GRANT ROLE sales_ro TO ROLE sales_rw;
```

5. 若要將 sales_rw 角色指派給業務工程師使用者，請使用下列範例。

```
GRANT ROLE sales_rw TO salesengineer;
```

步驟 6：以具有繼承唯讀角色的使用者身分查詢資料

在此步驟中，sales工程師使用者會在授與讀取權限之前會試圖更新事件表格。

1. 以銷售工程師使用者身分 Connect 至資料庫。
2. 銷售工程師使用者可以成功地從銷售結構描述的事件表中讀取資料。若要從銷售結構描述的事件表中選取含有 eventid 1 的事件，請使用下列範例。

```
SELECT * FROM sales.events where eventid=1;

+---------+---------+-------+--------+-----------------+---------------------+
| eventid | venueid | catid | dateid |    eventname    |      starttime      |
+---------+---------+-------+--------+-----------------+---------------------+
|       1 |     305 |     8 |   1851 | Gotterdammerung | 2008-01-25 14:30:00 |
```

```
+---------+---------+-------+--------+----------------+--------------------+
```

3. 若要嘗試從行銷結構描述的事件表中選取所有項目，請使用下列範例。sales工程師使用者沒有行銷結構描述中的資料表的權限，因此此查詢將導致權限遭拒錯誤。若要從行銷結構描述中的事件表中選取資料，您必須授與 sales_rw 角色 SELECT 權限，對行銷結構描述中的事件表格。

```
SELECT * FROM marketing.events;

ERROR: permission denied for schema marketing
```

4. 若要嘗試更新 eventid 1 的事件名稱，請執行下列範例。這個範例會導致權限遭拒錯誤，因為 sales 工程師使用者只具有銷售結構描述中的事件資料表的選取權限。若要更新事件資料表，您必須將 sales_rw 角色權限授與「更新」。

```
UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;

ERROR: permission denied for relation events
```

步驟 7：將更新和插入權限授與讀寫角色

在此步驟中，您將更新和插入權限授與 sales_rw 角色。

1. 以 dbadmin 使用者的身分 Connect 至資料庫。
2. 若要將「更新」、「插入」和「刪除」權限授與 sales_rw 角色，請使用下列範例。

```
GRANT UPDATE, INSERT, ON ALL TABLES IN SCHEMA sales TO role sales_rw;
```

步驟 8：以讀寫使用者身分查詢資料

在此步驟中，sales工程師在授與插入和更新權限的角色後成功更新表格。接下來，銷售工程師嘗試分析和真空事件表，但沒有這樣做。

1. 以銷售工程師使用者身分 Connect 至資料庫。
2. 若要更新事件 ID 1 的事件名稱，請執行下列範例。

```
UPDATE sales.events
```

```
SET eventname = 'Comment event'
WHERE eventid = 1;
```

3. 若要檢視先前查詢中所做的變更，請使用下列範例，從 sales 結構描述的事件資料表中選取 eventid 1 的事件。

```
SELECT * FROM sales.events WHERE eventid=1;

+---------+---------+-------+--------+--------------+---------------------+
| eventid | venueid | catid | dateid |   eventname  |      starttime      |
+---------+---------+-------+--------+--------------+---------------------+
|       1 |     305 |     8 |   1851 | Comment event | 2008-01-25 14:30:00 |
+---------+---------+-------+--------+--------------+---------------------+
```

4. 若要分析 sales 結構描述中更新的事件表格，請使用下列範例。此範例會導致權限遭拒錯誤，因為 sales工程師使用者沒有必要的權限，也不是銷售結構描述中事件資料表的擁有者。若要分析事件資料表，您必須使用 GRANT 命令將 sales_rw 角色權限授與「分析」。如需有關 ANALYZE 指令的更多資訊，請參閱ANALYZE。

```
ANALYZE sales.events;

               ERROR: skipping "events" --- only table or database owner can analyze
```

5. 若要真空更新的事件表格，請使用下列範例。此範例會導致權限遭拒錯誤，因為 sales工程師使用者沒有必要的權限，也不是銷售結構描述中事件資料表的擁有者。若要真空事件表，您必須使用 GRANT 命令將 sales_rw 角色權限授予真空。如需 VACUUM 指令的更多資訊，請參閱〈〉VACUUM。

```
VACUUM sales.events;

ERROR: skipping "events" --- only table or database owner can vacuum it
```

步驟 9：以管理員使用者身分分析資料庫中的資料表並加以真空

在此步驟中，dbadmin 用戶分析並吸塵所有表格。使用者擁有此資料庫的管理員權限，因此他們能夠執行這些命令。

1. 以 dbadmin 使用者的身分 Connect 至資料庫。
2. 若要分析 sales 結構描述中的事件表格，請使用下列範例。

```
ANALYZE sales.events;
```

3. 要真空銷售模式中的事件表，請使用下面的示例。

```
VACUUM sales.events;
```

4. 若要分析行銷結構描述中的事件表格，請使用下列範例。

```
ANALYZE marketing.events;
```

5. 若要真空行銷結構描述中的事件表格，請使用下列範例。

```
VACUUM marketing.events;
```

步驟 10：以讀寫使用者的身分截斷資料表

在此步驟中，sales工程師使用者會試圖截斷銷售結構描述中的事件資料表格，但只有在 dbadmin 使用者授與截斷權限時才會成功。

1. 以銷售工程師使用者身分 Connect 至資料庫。
2. 若要嘗試刪除 sales 結構描述中的事件資料表中的所有資料列，請使用下列範例。這個範例會導致錯誤，因為 sales工程師使用者沒有必要的權限，也不是銷售結構描述中事件資料表的擁有者。若要截斷事件資料表，您必須使用 GRANT 命令將 sales_rw 角色權限授與截斷。如需 TRUNCATE 命令的相關資訊，請參閱 TRUNCATE。

```
TRUNCATE sales.events;

ERROR: must be owner of relation events
```

3. 以 dbadmin 使用者的身分 Connect 至資料庫。
4. 若要將截斷資料表權限授與 sales_rw 角色，請使用下列範例。

```
GRANT TRUNCATE TABLE TO role sales_rw;
```

5. 使用查詢編輯器 v2，以銷售工程師使用者身分 Connect 線至資料庫。
6. 若要讀取 sales 結構描述中事件資料表中的前 10 個事件，請使用下列範例。

```
SELECT * FROM sales.events ORDER BY eventid LIMIT 10;
```

```
+---------+---------+-------+--------+----------------------------
+---------------------+
| eventid | venueid | catid | dateid |          eventname          |      starttime
    |
+---------+---------+-------+--------+----------------------------
+---------------------+
|       1 |     305 |     8 |   1851 | Comment event               | 2008-01-25
 14:30:00 |
|       2 |     306 |     8 |   2114 | Boris Godunov               | 2008-10-15
 20:00:00 |
|       3 |     302 |     8 |   1935 | Salome                      | 2008-04-19
 14:30:00 |
|       4 |     309 |     8 |   2090 | La Cenerentola (Cinderella) | 2008-09-21
 14:30:00 |
|       5 |     302 |     8 |   1982 | Il Trovatore                | 2008-06-05
 19:00:00 |
|       6 |     308 |     8 |   2109 | L Elisir d Amore            | 2008-10-10
 19:30:00 |
|       7 |     309 |     8 |   1891 | Doctor Atomic               | 2008-03-06
 14:00:00 |
|       8 |     302 |     8 |   1832 | The Magic Flute             | 2008-01-06
 20:00:00 |
|       9 |     308 |     8 |   2087 | The Fly                     | 2008-09-18
 19:30:00 |
|      10 |     305 |     8 |   2079 | Rigoletto                   | 2008-09-10
 15:00:00 |
+---------+---------+-------+--------+----------------------------
+---------------------+
```

7. 若要截斷 sales 結構描述中的事件表格，請使用下列範例。

```
TRUNCATE sales.events;
```

8. 若要從 sales 結構描述中更新的事件表中讀取資料，請使用下列範例。

```
SELECT * FROM sales.events ORDER BY eventid LIMIT 10;

+---------+---------+-------+--------+----------------------------
+---------------------+
| eventid | venueid | catid | dateid |          eventname          |      starttime
    |
```

```
+---------+---------+-------+--------+----------------------------
+---------------------+
```

為行銷結構描述建立唯讀和讀寫角色 (選用)

在此步驟中，您會為行銷結構描述建立唯讀和讀寫角色。

1. 以 dbadmin 使用者的身分 Connect 至資料庫。
2. 若要為行銷結構描述建立唯讀和讀寫角色，請使用下列範例。

```
CREATE ROLE marketing_ro;

CREATE ROLE marketing_rw;

GRANT USAGE ON SCHEMA marketing TO ROLE marketing_ro, ROLE marketing_rw;

GRANT SELECT ON ALL TABLES IN SCHEMA marketing TO ROLE marketing_ro;

GRANT ROLE marketing_ro TO ROLE marketing_rw;

GRANT INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA marketing TO ROLE marketing_rw;

CREATE USER marketinganalyst PASSWORD 'Test12345';

CREATE USER marketingengineer PASSWORD 'Test12345';

GRANT ROLE marketing_ro TO marketinganalyst;

GRANT ROLE marketing_rw TO marketingengineer;
```

適用於 RBAC 的系統功能 (選配)

Amazon Redshift 具有兩個功能，可以提供有關其他群組或角色中的使用者成員資格和角色成員資格的系統資訊：這些功能適用於超級用戶和普通用戶。超級使用者可以檢查所有角色成員資格。一般使用者只能檢查他們已被授與存取權的角色的成員資格。

若要使用角色成員函數

1. 以銷售工程師使用者身分 Connect 至資料庫。
2. 若要檢查 sales_rw 角色是否為 sales_ro 角色的成員，請使用下列範例。

```
SELECT role_is_member_of('sales_rw', 'sales_ro');

+-------------------+
| role_is_member_of |
+-------------------+
| true              |
+-------------------+
```

3. 若要檢查 sales_ro 角色是否為 sales_rw 角色的成員，請使用下列範例。

```
SELECT role_is_member_of('sales_ro', 'sales_rw');

+-------------------+
| role_is_member_of |
+-------------------+
| false             |
+-------------------+
```

若要使用使用者的成員函數

1. 以銷售工程師使用者身分 Connect 至資料庫。
2. 下列範例會嘗試檢查 salesAnalyst 使用者的使用者成員資格。此查詢會導致錯誤，因為業務工程師無法存取銷售分析員。若要成功執行此命令，請以 salesAnalyst 使用者身分連線至資料庫，並使用範例。

```
SELECT user_is_member_of('salesanalyst', 'sales_ro');

ERROR
```

3. 以超級使用者身分 Connect 至資料庫。
4. 若要在以超級使用者身分連線時檢查 salesAnalyst 使用者的成員資格，請使用下列範例。

```
SELECT user_is_member_of('salesanalyst', 'sales_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| true              |
+-------------------+
```

5. 以 dbadmin 使用者的身分 Connect 至資料庫。
6. 若要檢查銷售工程師使用者的成員資格，請使用下列範例。

```
SELECT user_is_member_of('salesengineer', 'sales_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| true              |
+-------------------+

SELECT user_is_member_of('salesengineer', 'marketing_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| false             |
+-------------------+

SELECT user_is_member_of('marketinganalyst', 'sales_ro');

+-------------------+
| user_is_member_of |
+-------------------+
| false             |
+-------------------+
```

RBAC 的系統檢視表 (選擇性)

若要檢視角色、角色指派給使用者、角色階層，以及透過角色對資料庫物件的權限，請使用 Amazon Redshift 的系統檢視。這些檢視可供超級使用者和一般使用者使用。超級用戶可以檢查所有角色詳細信息。一般使用者只能檢查他們已被授予存取權的角色的詳細資料。

1. 若要檢視叢集中明確授與角色的使用者清單，請使用下列範例。

```
SELECT * FROM svv_user_grants;
```

2. 若要檢視叢集中明確授與角色的角色清單，請使用下列範例。

```
SELECT * FROM svv_role_grants;
```

如需系統檢視的完整清單，請參閱SVV 中繼資料檢視。

搭配 RBAC 使用列層級安全性 (選用)

若要對敏感資料進行精細的存取控制，請使用資料列層級安全性 (RLS)。如需 RLS 的詳細資訊，請參閱 資料列層級安全性。

在此段落中，您會建立 RLS 原則，讓salesengineer使用者僅檢視資料cat表中具有美國職棒大聯盟catdesc值之資料列的權限。然後您以salesengineer使用者的身分查詢資料庫。

1. 以使用者身分 Connect 至資salesengineer料庫。
2. 若要檢視cat表格中的前 5 個項目，請使用下列範例。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+---------------------------------+
| catid | catgroup | catname |             catdesc             |
+-------+----------+---------+---------------------------------+
|     1 | Sports   | MLB     | Major League Baseball           |
|     2 | Sports   | NHL     | National Hockey League          |
|     3 | Sports   | NFL     | National Football League        |
|     4 | Sports   | NBA     | National Basketball Association |
|     5 | Sports   | MLS     | Major League Soccer             |
+-------+----------+---------+---------------------------------+
```

3. 以使用者身分 Connect 至資dbadmin料庫。
4. 若要為cat表格中的catdesc資料欄建立 RLS 原則，請使用下列範例。

```
CREATE RLS POLICY policy_mlb_engineer
WITH (catdesc VARCHAR(50))
USING (catdesc = 'Major League Baseball');
```

5. 若要將 RLS 原則附加至sales_rw角色，請使用下列範例。

```
ATTACH RLS POLICY policy_mlb_engineer ON sales.cat TO ROLE sales_rw;
```

6. 若要變更表格以開啟 RLS，請使用下列範例。

```
ALTER TABLE sales.cat ROW LEVEL SECURITY ON;
```

7. 以使用者身分 Connect 至資salesengineer料庫。
8. 若要嘗試檢視cat表格中的前 5 個項目，請使用下列範例。請注意，只有catdesc欄位為時，才會顯示項目Major League Baseball。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+----------------------+
| catid | catgroup | catname |        catdesc       |
+-------+----------+---------+----------------------+
|     1 | Sports   | MLB     | Major League Baseball |
+-------+----------+---------+----------------------+
```

9. 以使用者身分 Connect 至資salesanalyst料庫。
10. 若要嘗試檢視cat表格中的前 5 個項目，請使用下列範例。請注意，由於已套用預設拒絕所有策略，因此不會顯示任何項目。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+----------------------+
| catid | catgroup | catname |        catdesc       |
+-------+----------+---------+----------------------+
```

11. 以使用者身分 Connect 至資dbadmin料庫。
12. 若要將 IGNORE RLS 權限授與sales_ro角色，請使用下列範例。這會授與salesanalyst使用者忽略 RLS 原則的權限，因為他們是sales_ro角色的成員。

```
GRANT IGNORE RLS TO ROLE sales_ro;
```

13. 以使用者身分 Connect 至資salesanalyst料庫。
14. 若要檢視cat表格中的前 5 個項目，請使用下列範例。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+---------------------------------+
| catid | catgroup | catname |             catdesc             |
+-------+----------+---------+---------------------------------+
|     1 | Sports   | MLB     | Major League Baseball           |
|     2 | Sports   | NHL     | National Hockey League          |
|     3 | Sports   | NFL     | National Football League        |
|     4 | Sports   | NBA     | National Basketball Association |
|     5 | Sports   | MLS     | Major League Soccer             |
+-------+----------+---------+---------------------------------+
```

15以使用者身分 Connect 至資dbadmin料庫。

16若要撤銷sales_ro角色的 IGNORE RLS 權限，請使用下列範例。

```
REVOKE IGNORE RLS FROM ROLE sales_ro;
```

17以使用者身分 Connect 至資salesanalyst料庫。

18若要嘗試檢視cat表格中的前 5 個項目，請使用下列範例。請注意，由於已套用預設拒絕所有策略，因此不會顯示任何項目。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+----------------------+
| catid | catgroup | catname |        catdesc       |
+-------+----------+---------+----------------------+
```

19以使用者身分 Connect 至資dbadmin料庫。

20若要從cat表格中斷連結 RLS 原則，請使用下列範例。

```
DETACH RLS POLICY policy_mlb_engineer ON cat FROM ROLE sales_rw;
```

21以使用者身分 Connect 至資salesanalyst料庫。

22若要嘗試檢視cat表格中的前 5 個項目，請使用下列範例。請注意，由於已套用預設拒絕所有策略，因此不會顯示任何項目。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;

+-------+----------+---------+---------------------------------+
| catid | catgroup | catname |             catdesc             |
+-------+----------+---------+---------------------------------+
|     1 | Sports   | MLB     | Major League Baseball           |
|     2 | Sports   | NHL     | National Hockey League          |
|     3 | Sports   | NFL     | National Football League        |
|     4 | Sports   | NBA     | National Basketball Association |
|     5 | Sports   | MLS     | Major League Soccer             |
+-------+----------+---------+---------------------------------+
```

23以使用者身分 Connect 至資dbadmin料庫。

24若要刪除 RLS 原則，請使用下列範例。

```
DROP RLS POLICY policy_mlb_engineer;
```

25若要移除 RLS，請使用下列範例。

```
ALTER TABLE cat ROW LEVEL SECURITY OFF;
```

相關主題

如需有關 RBAC 的詳細資訊，請參閱下列文件：

- 角色階層
- 角色指派
- 資料庫物件許可
- RBAC 的 ALTER DEFAULT PRIVILEGES

資料列層級安全性

使用 Amazon Redshift 中的資料列層級安全 (RLS)，您可以對敏感資料進行精細的存取控制。您可以根據資料庫物件層級定義的安全政策，決定哪些使用者或角色可以存取結構描述或資料表內的特定資料記錄。除了可用來授予使用者資料欄子集許可的資料欄層級安全之外，還可以使用 RLS 政策進一步限制可見資料欄的特定資料列存取。如需資料欄層級安全的相關資訊，請參閱 欄位層級存取控制的使用須知。

在資料表上強制執行 RLS 政策時，您可以在使用者執行查詢時限制傳回的結果集。

建立 RLS 政策時，您可以指定運算式來指出 Amazon Redshift 是否在查詢中傳回資料表中的任何現有資料列。藉由建立 RLS 政策來限制存取，您不必在查詢中新增或外部化其他條件。

建立 RLS 政策時，建議您建立簡單的政策，並避免在政策中使用複雜的陳述式。定義 RLS 政策時，請勿在以政策為基礎的政策定義中使用過多的資料表聯結。

當政策參照查閱資料表時，除了政策所在的資料表之外，Amazon Redshift 還會掃描其他表格。對於已附加 RLS 政策的使用者，以及未附加任何政策的使用者，相同查詢之間會有效能差異。

在 SQL 陳述式中使用 RLS 政策

在 SQL 陳述式中使用 RLS 政策時，Amazon Redshift 會套用下列規則：

- 根據預設，Amazon Redshift 會將 RLS 政策套用至 SELECT、UPDATE 和 DELETE 陳述式。
- 對於 SELECT 和 UNLOAD，Amazon Redshift 會根據您定義的政策篩選資料列。
- 對於 UPDATE，Amazon Redshift 只會更新您可以看到的資料列。如果政策限制了資料表中的資料列子集，您就無法將其更新。
- 對於 DELETE，您只能刪除您可以看到的資料列。如果政策限制了資料表中的資料列子集，您就無法將其刪除。對於 TRUNCATE，您仍然可以截斷資料表。
- 對於 CREATE TABLE LIKE，使用 LIKE 選項建立的資料表不會繼承從來源資料表設定的許可。同樣地，目標資料表不會繼承來源資料表的 RLS 政策。

每個使用者結合多個政策

Amazon Redshift 中的 RLS 支援為每個使用者和物件附加多個政策。為使用者定義多個政策時，Amazon Redshift 會使用 AND 或 OR 語法 (取決於資料表的 RLS CONJUNCTION TYPE 設定) 套用所有政策。如需結合類型的更多相關資訊，請參閱 ALTER TABLE。

資料表上的多個政策都可以與您建立關聯。您可以直接附加多個政策，或者您屬於多個角色，而這些角色附加了不同的政策。

當多個政策應限制指定關係中的資料列存取時，您可以將關係的 RLS CONJUNCTION TYPE 設定為 AND。請考量下列範例。Alice 只能看到具有 NBA「貓名」的體育賽事做為指定政策。

```
-- Create an analyst role and grant it to a user named Alice.
CREATE ROLE analyst;
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
GRANT ROLE analyst TO alice;

-- Create an RLS policy that only lets the user see sports.
CREATE RLS POLICY policy_sports
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Sports');

-- Create an RLS policy that only lets the user see NBA.
CREATE RLS POLICY policy_nba
WITH (catname VARCHAR(10))
USING (catname = 'NBA');

-- Attach both to the analyst role.
ATTACH RLS POLICY policy_sports ON category TO ROLE analyst;
ATTACH RLS POLICY policy_nba ON category TO ROLE analyst;

-- Activate RLS on the category table with AND CONJUNCTION TYPE.
ALTER TABLE category ROW LEVEL SECURITY ON CONJUNCTION TYPE AND;

-- Change session to Alice.
SET SESSION AUTHORIZATION alice;

-- Select all from the category table.
SELECT catgroup, catname
FROM category;

 catgroup | catname
---------+---------
 Sports   | NBA
(1 row)
```

當多個策略應允許使用者查看指定關係中的更多資料列時，使用者可以將關係的 RLS CONJUNCTION TYPE 設定為 OR。請考量下列範例。Alice 只能看到「音樂會」和「體育」做為指定政策。

```
-- Create an analyst role and grant it to a user named Alice.
CREATE ROLE analyst;
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
GRANT ROLE analyst TO alice;

-- Create an RLS policy that only lets the user see concerts.
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');

-- Create an RLS policy that only lets the user see sports.
CREATE RLS POLICY policy_sports
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Sports');

-- Attach both to the analyst role.
ATTACH RLS POLICY policy_concerts ON category TO ROLE analyst;
ATTACH RLS POLICY policy_sports ON category TO ROLE analyst;

-- Activate RLS on the category table with OR CONJUNCTION TYPE.
ALTER TABLE category ROW LEVEL SECURITY ON CONJUNCTION TYPE OR;

-- Change session to Alice.
SET SESSION AUTHORIZATION alice;

-- Select all from the category table.
SELECT catgroup, count(*)
FROM category
GROUP BY catgroup ORDER BY catgroup;

 catgroup | count
---------+-------
 Concerts |  3
 Sports   |  5
(2 rows)
```

RLS 政策擁有權和管理

身為擁有 sys:secadmin 角色的超級使用者、安全性管理員或使用者，您可以建立、修改或管理資料表的所有 RLS 政策。在物件層級上，您可以開啟或關閉資料列層級安全，而不必修改資料表的結構描述定義。

若要開始使用資料列層級安全，下列是您可以使用的 SQL 陳述式：

- 使用 ALTER TABLE 陳述式來開啟或關閉資料表上的 RLS。如需詳細資訊，請參閱 ALTER TABLE。
- 使用 CREATE RLS POLICY 陳述式為一或多個資料表建立安全政策，並在政策中指定一或多個使用者或角色。

  如需詳細資訊，請參閱 CREATE RLS POLICY。
- 使用 ALTER RLS POLICY 陳述式來修改政策，例如變更政策定義。您可以針對多個資料表或檢視使用相同的政策。

  如需詳細資訊，請參閱 ALTER RLS POLICY。
- 使用 ATTACH RLS POLICY 陳述式可將政策附加至一或多個關係、一或多個使用者或角色。

  如需詳細資訊，請參閱 ATTACH RLS POLICY 。
- 使用 DETACH RLS 原則陳述式可將原則從一或多個關係、一或多個使用者或角色中分離。

  如需詳細資訊，請參閱 DETACH RLS POLICY 。
- 您可以使用 DROP RLS POLICY 陳述式來捨棄政策。

  如需詳細資訊，請參閱 DROP RLS POLICY 。
- 您可以使用 GRANT 和 REVOKE 陳述式，明確授予及撤銷參考查閱資料表之 RLS 政策的 SELECT 許可。如需詳細資訊，請參閱 GRANT 及 REVOKE。

若要監控所建立的證測，sys:secadmin 可以檢視 SVV_RLS_POLICY 和 SVV_RLS_ATTACHED_POLICY。

若要列出受 RS 保護的關係，sys:secadmin 可以檢視 SVV_RLS_RELATION。

若要追蹤參照受 RLS 保護關係之查詢上的 RLS 政策應用程式，超級使用者、sys:operator 或任何具有系統許可 ACCESS SYSTEM TABLE 的使用者都可以檢視 SVV_RLS_APPLIED_POLICY 。請注意，在預設情況下，sys:secadmin 不會授予這些許可。

若要查詢附加 RLS 政策的資料表，但看不到這些資料表，您可以將 IGNORE RLS 許可授予任何使用者。身為超級使用者或 sys:secadmin 的使用者會自動獲得 IGNORE RLS 許可。如需詳細資訊，請參閱 GRANT。

若要說明 EXPLAIN 計劃中查詢的 RLS 政策篩選，以針對與 RLS 相關的查詢進行疑難排解，您可以將 EXPLAIN RLS 許可授予任何使用者。如需詳細資訊，請參閱 GRANT 及 EXPLAIN。

政策相依物件和原則

為了提供應用程式的安全性，並防止政策物件過時或無效，Amazon Redshift 不允許捨棄或變更 RLS 政策所參照的物件。

下列範例說明結構描述相依性的追蹤方式。

```
-- The CREATE and ATTACH policy statements for `policy_events` references some
-- target and lookup tables.
-- Target tables are tickit_event_redshift and target_schema.target_event_table.
-- Lookup table is tickit_sales_redshift.
-- Policy `policy_events` has following dependencies:
--   table tickit_sales_redshift column eventid, qtysold
--   table tickit_event_redshift column eventid
--   table target_event_table column eventid
--   schema public and target_schema
CREATE RLS POLICY policy_events
WITH (eventid INTEGER)
USING (
    eventid IN (SELECT eventid FROM tickit_sales_redshift WHERE qtysold <3)
);

ATTACH RLS POLICY policy_events ON tickit_event_redshift TO ROLE analyst;

ATTACH RLS POLICY policy_events ON target_schema.target_event_table TO ROLE consumer;
```

以下列出 Amazon Redshift 針對 RLS 政策追蹤的結構描述物件相依性。

- 追蹤目標資料表的結構描述物件相依性時，Amazon Redshift 會遵循下列規則：
  - 當您捨棄目標資料表時，Amazon Redshift 會將政策與關係、使用者、角色或公開分離。
  - 當您重新命名目標資料表名稱時，附加的政策不會受到影響。
  - 如果您先將政策捨棄或分離，您只能捨棄政策定義中參照的目標資料表資料欄。這也適用於指定 CASCADE 選項時。您可以捨棄目標資料表中的其他資料欄。
  - 您無法重命名目標資料表的參照資料欄。若要重新命名參照的資料欄，請先將政策分離。這也適用於指定 CASCADE 選項時。
  - 即使您指定 CASCADE 選項，也無法變更參照資料欄的類型。
- 追蹤查閱資料表的結構描述物件相依性時，Amazon Redshift 會遵循下列規則：
  - 您無法捨棄查閱資料表。若要捨棄查閱資料表，請先捨棄參照查詢資料表的政策。

  - 您無法重新命名查閱資料表。若要重新命名查閱資料表，請先捨棄參照查詢資料表的政策。這也適用於指定 CASCADE 選項時。
  - 您無法捨棄政策定義中使用的查閱資料表資料欄。若要捨棄政策定義中使用的查閱資料表資料欄，請先捨棄參照查詢資料表的政策。這也適用於在 ALTER TABLE DROP COLUMN 陳述式中指定 CASCADE 選項時。您可以捨棄查閱資料表中的其他資料欄。
  - 您無法重命名查閱資料表的參照資料欄。若要重新命名參照的資料欄，請先捨棄參照查詢資料表的政策。這也適用於指定 CASCADE 選項時。
  - 您無法變更參照資料欄的類型。
- 捨棄使用者或角色時，Amazon Redshift 會自動分離附加到該使用者或角色的所有政策。
- 當您在 DROP SCHEMA 陳述式中使用 CASCADE 選項時，Amazon Redshift 也會捨棄結構描述中的關係。還會捨棄任何其他結構描述中依賴於已捨棄結構描述中關係的關係。對於政策中作為查閱資料表的關係，Amazon Redshift 會使 DROP SCHEMA DDL 失敗。對於 DROP SCHEMA 陳述式捨棄的任何關係，Amazon Redshift 會將所有附加到這些關係的政策分離。
- 您只能在捨棄政策時捨棄查詢函數 (政策定義中參照的函數)。這也適用於指定 CASCADE 選項時。
- 將政策附加至資料表時，Amazon Redshift 會檢查此資料表是否為不同政策中的查閱資料表。如果是這種情況，Amazon Redshift 將不允許在此資料表中附加政策。
- 建立 RLS 政策時，Amazon Redshift 會檢查此資料表是否為任何其他 RLS 政策的目標資料表。如果是這種情況，Amazon Redshift 將不允許在此資料表上建立政策。

使用 RLS 政策的考量

以下是使用 RLS 政策時的考量事項：

- Amazon Redshift 會將 RLS 政策套用至 SELECT、UPDATE 或 DELETE 陳述式。
- Amazon Redshift 不會將 RLS 政策套用至 INSERT、COPY、ALTER TABLE APPEND 陳述式。
- 資料列層級安全可搭配資料欄層級安全來保護您的資料。
- 如果您的 Amazon Redshift 叢集原本使用的是支援 RLS 的最新可用版本，但降級為舊版，當您在附加 RLS 政策的基本資料表上執行查詢時，Amazon Redshift 會傳回錯誤訊息。sys:secadmin 可以對被授予限制政策的使用者撤銷存取權、關閉資料表上的 RLS，以及捨棄政策。
- 針對來源關係開啟 RLS 時，Amazon Redshift 會針對超級使用者、已明確授予系統許可 IGNORE RLS 的使用者或 sys:secadmin 角色支援 ALTER TABLE APPEND 陳述式。在這種情況下，您可以執行 ALTER TABLE APPEND 陳述式，藉由從現有來源資料表移動資料來將資料列附加至目標資料表。Amazon Redshift 會將所有元組從來源關係移動到目標關係。目標關係的 RLS 狀態不會影響 ALTER TABLE APPEND 陳述式。

- 若要協助從其他資料倉儲系統移轉，您可以指定變數名稱和值，以設定和擷取用於連線的自訂工作階段內容變數。

  下列範例會針對資料列層級安全 (RLS) 政策設定工作階段內容變數。

```
-- Set a customized context variable.
SELECT set_config('app.category', 'Concerts', FALSE);

-- Create a RLS policy using current_setting() to get the value of a customized
 context variable.
CREATE RLS POLICY policy_categories
WITH (catgroup VARCHAR(10))
USING (catgroup = current_setting('app.category', FALSE));

-- Set correct roles and attach the policy on the target table to one or more roles.
ATTACH RLS POLICY policy_categories ON tickit_category_redshift TO ROLE analyst, ROLE
 dbadmin;
```

  如需有關如何設定和擷取自訂工作階段內容變數的詳細資訊，請參閱 SET、SET_CONFIG、SHOW、CURRENT_SETTING 和 RESET。

- 在 DECLARE 和 FETCH 之間或後續 FETCH 陳述式之間使用 SET SESSION AUTHORIZATION 變更工作階段使用者，將不會根據 DECLARE 時間的使用者政策重新整理已就緒的計劃。當游標與受 RLS 保護的資料表搭配使用時，請避免變更工作階段使用者。
- 當檢視物件中的基底物件受到 RLS 保護時，附加至執行查詢之使用者的政策會套用至個別的基底物件。這與物件層級許可檢查不同，檢視擁有者的許可會根據檢視基礎物件進行檢查。您可以在查詢的 EXPLAIN 計畫輸出中檢視受到 RS 保護的關係。
- 在附加至使用者的關係的 RLS 政策中參照使用者定義函數 (UDF) 時，使用者必須擁有 UDF 的 EXECUTE 許可，才能查詢關係。
- 資料列層級安全性可能會限制最佳查詢效果。在大型資料集上部署受 RLS 保護的視觀表之前，我們建議您仔細評估查詢效能。
- 套用至最新繫結視觀表的資料列層級安全政策，可能會推入聯合資料表。這些 RLS 政策可能會顯示在外部處理引擎日誌檔中。

限制

下列是使用 RLS 政策時的限制：

- Amazon Redshift 可針對具有複雜聯結的特定 RLS 政策支援 SELECT 陳述式，但不支援 UPDATE 或 DELETE 陳述式。如果使用 UPDATE 或 DELETE，Amazon Redshift 會傳回以下錯誤：

```
ERROR: One of the RLS policies on target relation is not supported in UPDATE/DELETE.
```

- 每當在附加至使用者的關係的 RLS 政策中參照使用者定義函數 (UDF) 時，使用者必須擁有 UDF 的 EXECUTE 許可，才能查詢關係。
- 不支援相關子查詢。Amazon Redshift 會傳回以下錯誤：

```
ERROR: RLS policy could not be rewritten.
```

- RLS 政策無法附加到外部資料表和具體化視觀表。
- Amazon Redshift 不支援使用 RLS 進行資料共用。如果關係沒有關閉用於資料共用的 RLS，則取用者叢集上的查詢會失敗，並出現下列錯誤：

```
RLS-protected relation "rls_protected_table" cannot be accessed via datasharing
 query.
```

- 在跨資料庫查詢中，Amazon Redshift 會阻止您讀取受 RLS 保護的關係。具有 IGNORE RLS 許可的使用者可以使用跨資料庫查詢存取受保護的關係。當沒有 IGNORE RLS 許可的使用者透過跨資料庫查詢存取受 RLS 保護的關係時，會出現下列錯誤：

```
RLS-protected relation "rls_protected_table" cannot be accessed via cross-database
 query.
```

- ALTER RLS POLICY 只支援 USING (using_predicate_exp) 子句修改 RLS 政策。執行 ALTER RLS POLICY 時，您無法使用 WITH 子句修改 RLS 政策。
- 如果下列任一組態選項的值不符合工作階段的預設值，您就無法查詢已開啟資料列層級安全的關係：
  - `enable_case_sensitive_super_attribute`
  - `enable_case_sensitive_identifier`
  - `downcase_delimited_identifier`

  如果您嘗試查詢已開啟資料列層級安全的關係，並看到「受 RLS 保護的關係不支援工作階段層級組太，因為區分大小寫設定與其預設值不同」的訊息，請考慮重設工作階段的組態選項。
- 當您佈建的叢集或無伺服器命名空間具有任何資料列層級安全性政策時，一般使用者會無法使用下列命令：

```
ALTER <current_user> SET enable_case_sensitive_super_attribute/
enable_case_sensitive_identifier/downcase_delimited_identifier
```

建立 RLS 政策時，建議您變更一般使用者的預設組態選項設定，以符合建立政策時的工作階段組態選項設定。超級使用者和擁有 ALTER USER 權限的使用者可以使用參數群組設定或 ALTER USER 命令來執行此操作。如需有關參數群組的詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 參數群組。如需 ALTER USER 命令的相關資訊，請參閱 ALTER USER。

- 使用 CREATE VIEW 命令的一般使用者，無法取代具有資料列層級安全政策的視觀表和近期繫結視觀表。若要以 RLS 政策取代視觀表或 LBV，請先卸離任何附加的 RLS 政策、取代視觀表或 LBV，然後重新附加政策。具有 `sys:secadmin permission` 的超級使用者和一般使用者都可以在具有 RLS 政策，但未卸離政策的視觀表或 LBV 上使用 CREATE VIEW。
- 具有資料列層級安全政策的視觀表無法參考系統資料表和系統視觀表。
- 一般視觀表所參考的近期繫結視觀表，不得為受 RLS 保護。
- RLS 保護的關係和來自資料湖的巢狀資料，均無法在相同的查詢中存取。

RLS 效能的最佳實務

以下最佳實務可確保 Amazon Redshift 在受 RLS 保護的資料表上獲得更好的效能。

運算子和函數的安全性

查詢受 RLS 保護的資料表時，使用某些運算子或函數可能會導致效能降低。Amazon Redshift 會將運算子和函數分類為在查詢受 RLS 保護的資料表時是安全或不安全的。當函數或運算子沒有因為輸入而有任何可觀察到的副作用時，函數或運算子就會被歸類為 RLS 安全。特別是，RLS 安全函數或運算子不能是以下其中一項：

- 輸出一個輸入值，或任何依賴於輸入值的值，無論有沒有錯誤訊息。
- 失敗或傳回依賴於輸入值的錯誤。

RLS 不安全的運算子包括：

- 算術運算子 — +、-、/、*、%。
- 文字運算子 — LIKE 和 SIMILAR TO。
- 轉換運算子。

- UDF。

使用下列 SELECT 陳述式來檢查運算子和函數的安全性。

```
SELECT proname, proc_is_rls_safe(oid) FROM pg_proc;
```

在受 RLS 保護的資料表上規劃查詢時，Amazon Redshift 會對包含 RLS 不安全運算子和函數的使用者述詞的評估順序施加限制。查詢受 RLS 保護的資料表時，參照 RLS 不安全運算子或函數的查詢可能會導致效能降低。當 Amazon Redshift 無法將 RLS 不安全述詞推送至基底資料表掃描以利用排序索引鍵時，效能可能會大幅降低。為了獲得更好的效能，請避免使用利用排序索引鍵的 RLS 不安全述詞進行查詢。若要確認 Amazon Redshift 能夠向下推送運算子和函數，您可以將 EXPLAIN 陳述式與系統許可 EXPLACE RLS 結合使用。

結果快取

為了縮短查詢執行期並改善系統效能，Amazon Redshift 會將特定查詢類型的結果快取在領導者節點的記憶體中。

當未受保護資料表的所有條件皆成立，且下列所有條件皆成立時，Amazon Redshift 會在新查詢掃描受 RLS 保護的資料表時使用快取結果：

- 政策中的資料表或檢視尚未經過修改。
- 政策並未使用每次執行時都必須求值的函數，例如 GETDATE 或 CURRENT_USER。

為了獲得更好的效能，請避免使用不符合前述條件的政策述詞。

如需 Amazon Redshift 中結果快取的詳細資訊，請參閱 結果快取 。

複雜政策

為了獲得更好的效能，請避免在聯結多個資料表的子查詢中使用複雜的政策。

建立、附加、分離及捨棄 RLS 政策

您可以執行下列動作：

- 若要建立 RLS 政策，請使用 CREATE RLS POLICY 命令。
- 若要將資料表上的 RLS 政策附加至一或多個使用者或角色，請使用 ATTACH RLS POLICY 命令。

- 若要將資料表上的資料列層級安全政策從一或多個使用者或角色中分離，請使用 DETACH RLS POLICY 命令。
- 若要捨棄所有資料庫中所有資料表的 RLS 政策，請使用 DROP RLS POLICY 命令。

以下是說明超級使用者如何建立某些使用者和角色的 end-to-end 範例。然後，具有 secadmin 角色的使用者會建立、附加、分離和捨棄 RLS 政策。此範例會使用票卷範例資料庫。如需詳細資訊，請參閱《Amazon Redshift 入門指南》中的將資料從 Amazon S3 載入到 Amazon Redshift。

```
-- Create users and roles referenced in the policy statements.
CREATE ROLE analyst;
CREATE ROLE consumer;
CREATE ROLE dbadmin;
CREATE ROLE auditor;
CREATE USER bob WITH PASSWORD 'Name_is_bob_1';
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
CREATE USER joe WITH PASSWORD 'Name_is_joe_1';
CREATE USER molly WITH PASSWORD 'Name_is_molly_1';
CREATE USER bruce WITH PASSWORD 'Name_is_bruce_1';
GRANT ROLE sys:secadmin TO bob;
GRANT ROLE analyst TO alice;
GRANT ROLE consumer TO joe;
GRANT ROLE dbadmin TO molly;
GRANT ROLE auditor TO bruce;
GRANT ALL ON TABLE tickit_category_redshift TO PUBLIC;
GRANT ALL ON TABLE tickit_sales_redshift TO PUBLIC;
GRANT ALL ON TABLE tickit_event_redshift TO PUBLIC;

-- Create table and schema referenced in the policy statements.
CREATE SCHEMA target_schema;
GRANT ALL ON SCHEMA target_schema TO PUBLIC;
CREATE TABLE target_schema.target_event_table (LIKE tickit_event_redshift);
GRANT ALL ON TABLE target_schema.target_event_table TO PUBLIC;

-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check the tuples visible to analyst alice.
-- Should contain all 3 categories.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;
```

```
-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');

SELECT poldb, polname, polalias, polatts, polqual, polenabled, polmodifiedby FROM
 svv_rls_policy WHERE poldb = CURRENT_DATABASE();

ATTACH RLS POLICY policy_concerts ON tickit_category_redshift TO ROLE analyst, ROLE
 dbadmin;

ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;

SELECT * FROM svv_rls_attached_policy;

-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check that tuples with only `Concert` category will be visible to analyst alice.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to consumer joe.
SET SESSION AUTHORIZATION joe;

-- Although the policy is attached to a different role, no tuples will be
-- visible to consumer joe because the default deny all policy is applied.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to dbadmin molly.
SET SESSION AUTHORIZATION molly;

-- Check that tuples with only `Concert` category will be visible to dbadmin molly.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Check that EXPLAIN output contains RLS SecureScan to prevent disclosure of
-- sensitive information such as RLS filters.
```

```
EXPLAIN SELECT catgroup, count(*) FROM tickit_category_redshift GROUP BY catgroup ORDER
 BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

-- Grant IGNORE RLS permission so that RLS policies do not get applicable to role
 dbadmin.
GRANT IGNORE RLS TO ROLE dbadmin;

-- Grant EXPLAIN RLS permission so that anyone in role auditor can view complete
 EXPLAIN output.
GRANT EXPLAIN RLS TO ROLE auditor;

-- Change session to dbadmin molly.
SET SESSION AUTHORIZATION molly;

-- Check that all tuples are visible to dbadmin molly because `IGNORE RLS` is granted
 to role dbadmin.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to auditor bruce.
SET SESSION AUTHORIZATION bruce;

-- Check explain plan is visible to auditor bruce because `EXPLAIN RLS` is granted to
 role auditor.
EXPLAIN SELECT catgroup, count(*) FROM tickit_category_redshift GROUP BY catgroup ORDER
 BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

DETACH RLS POLICY policy_concerts ON tickit_category_redshift FROM ROLE analyst, ROLE
 dbadmin;

-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check that no tuples are visible to analyst alice.
-- Although the policy is detached, no tuples will be visible to analyst alice
-- because of default deny all policy is applied if the table has RLS on.
SELECT catgroup, count(*)
```

```
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

CREATE RLS POLICY policy_events
WITH (eventid INTEGER) AS ev
USING (
    ev.eventid IN (SELECT eventid FROM tickit_sales_redshift WHERE qtysold <3)
);

ATTACH RLS POLICY policy_events ON tickit_event_redshift TO ROLE analyst;
ATTACH RLS POLICY policy_events ON target_schema.target_event_table TO ROLE consumer;

RESET SESSION AUTHORIZATION;

-- Can not cannot alter type of dependent column.
ALTER TABLE target_schema.target_event_table ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_event_redshift ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_sales_redshift ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_sales_redshift ALTER COLUMN qtysold TYPE float;

-- Can not cannot rename dependent column.
ALTER TABLE target_schema.target_event_table RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_event_redshift RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_sales_redshift RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_sales_redshift RENAME COLUMN qtysold TO renamed_qtysold;

-- Can not drop dependent column.
ALTER TABLE target_schema.target_event_table DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_event_redshift DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_sales_redshift DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_sales_redshift DROP COLUMN qtysold CASCADE;

-- Can not drop lookup table.
DROP TABLE tickit_sales_redshift CASCADE;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

DROP RLS POLICY policy_concerts;
DROP RLS POLICY IF EXISTS policy_events;
```

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY OFF;

RESET SESSION AUTHORIZATION;

-- Drop users and roles.
DROP USER bob;
DROP USER alice;
DROP USER joe;
DROP USER molly;
DROP USER bruce;
DROP ROLE analyst;
DROP ROLE consumer;
DROP ROLE auditor FORCE;
DROP ROLE dbadmin FORCE;
```

中繼資料安全性

如同 Amazon Redshift 的資料列層級安全性一樣，中繼資料安全性可讓您更精細地控制中繼資料。如果針對已佈建的叢集或無伺服器工作群組啟用中繼資料安全性，則使用者可以查看其具有檢視存取權之物件的中繼資料。中繼資料安全性可讓您根據需求分開可見性。例如，您可以使用單一資料倉儲來集中所有資料儲存。但是，如果您為多個扇區儲存資料，則管理安全性會較為棘手。啟用中繼資料安全性後，可以設定可見性。一個扇區的使用者對其物件有更高可見性，同時可限制對另一個磁區使用者的檢視權。中繼資料安全性支援所有物件類型，例如結構描述、資料表、視觀表、具體化視觀表、預存程序、使用者定義函數和機器學習模型。

使用者可以在下列情況下查看物件的中繼資料：

- 如果將物件存取權授予使用者。
- 如果將物件存取權授予使用者所屬的群組或角色。
- 物件是公有的。
- 使用者是資料庫物件的擁有者。

若要啟用中繼資料安全性，請使用 ALTER SYSTEM 命令。以下是如何使用具有中繼資料安全性的 ALTER SYSTEM 命令的語法。

```
ALTER SYSTEM SET metadata_security=[true|t|on|false|f|off];
```

當您啟用中繼資料安全性時，擁有必要權限的所有使用者都有權查看他們可存取之物件的相關中繼資料。如果您只想讓特定使用者查看中繼資料安全性，請將 ACCESS CATALOG 權限授予某角色，然後再將該角色指派給使用者。如需有關使用角色進一步控制安全性的詳細資訊，請參閱角色型存取控制。

下列範例示範如何將 ACCESS CATALOG 權限授予某角色，然後再將該角色指派給使用者。如需有關授予權限的詳細資訊，請參閱 GRANT 命令。

```
CREATE ROLE sample_metadata_viewer;

GRANT ACCESS CATALOG TO ROLE sample_metadata_viewer;

GRANT ROLE sample_metadata_viewer to salesadmin;
```

如果您偏好使用已定義的角色，則系統定義的角色operator、secadmin、dba 和 superuser 所有角色都具有檢視物件中繼資料的必要權限。預設情況下，超級使用者可以查看完整的目錄。

```
GRANT ROLE operator to sample_user;
```

如果您使用角色來控制中繼資料安全性，則可以存取角色型存取控制隨附的所有系統檢視和功能。例如，您可以查詢 SVV_ROLES 檢視來查看所有角色。若要查看使用者是否為角色或群組的成員，請使用 USER_IS_MEMBER_OF 函數。如需 SVV 視觀表的完整清單，請參閱 SVV 中繼資料視觀表。如需系統資訊函數清單，請參閱系統資訊函數。

動態資料遮罩

概觀

使用 Amazon Redshift 中的動態資料遮罩 (DDM)，您可以保護資料倉儲中的敏感資料。您可以操控 Amazon Redshift 在查詢時向使用者顯示敏感資料的方式，而無需在資料庫中進行轉換。您可以透過將自訂混淆規則套用至指定使用者或角色的遮罩政策，來控制資料的存取。如此一來，您就可以回應不斷變更的隱私權需求，而不必修改基礎資料或編輯 SQL 查詢。

動態資料遮罩原則會隱藏、混淆或虛擬化符合指定格式的資料。當附加至資料表時，遮罩運算式會套用至其一或多個資料欄。您可以進一步修改遮罩政策，只將政策套用至特定使用者，或套用至可以使用角色型存取控制 (RBAC) 建立的使用者定義角色。此外，您可以在建立遮罩政策時使用條件資料欄，在儲存格層級上套用 DDM。如需條件式遮罩的詳細資訊，請參閱 條件式動態資料遮罩。

您可以將不同混淆層級的多個遮罩政策套用至資料表中的相同資料欄，並將期指派給不同的角色。當您有不同角色且搭配套用至一個資料欄的不同政策時，為避免發生衝突，您可以為每個應用程式設定優先

順序。如此一來，您就可以控制指定使用者或角色可以存取的資料。DDM 政策可以部分或完全修訂資料，或使用以 SQL、Python 或 AWS Lambda撰寫的使用者定義函數來雜湊資料。透過使用雜湊來遮罩資料，您可以在不存取潛在敏感資訊的情況下對此資料套用聯結。

電子nd-to-end 示例

下列 end-to-end 範例顯示如何建立遮罩原則並將其附加至資料欄。這些政策可讓使用者存取資料欄並查看不同的值，視附加至其角色的政策中的混淆程度而定。您必須是超級使用者或具有 `sys:secadmin` 角色才能執行此範例。

建立遮罩政策

首先，建立一個資料表，並填入信用卡值。

```
--create the table
CREATE TABLE credit_cards (
  customer_id INT,
  credit_card TEXT
);

--populate the table with sample values
INSERT INTO credit_cards
VALUES
  (100, '4532993817514842'),
  (100, '4716002041425888'),
  (102, '5243112427642649'),
  (102, '6011720771834675'),
  (102, '6011378662059710'),
  (103, '373611968625635')
;

--run GRANT to grant permission to use the SELECT statement on the table
GRANT SELECT ON credit_cards TO PUBLIC;

--create two users
CREATE USER regular_user WITH PASSWORD '1234Test!';

CREATE USER analytics_user WITH PASSWORD '1234Test!';

--create the analytics_role role and grant it to analytics_user
--regular_user does not have a role
CREATE ROLE analytics_role;
```

```
GRANT ROLE analytics_role TO analytics_user;
```

接下來，建立要套用至分析角色的遮罩政策。

```
--create a masking policy that fully masks the credit card number
CREATE MASKING POLICY mask_credit_card_full
WITH (credit_card VARCHAR(256))
USING ('000000XXXX0000'::TEXT);

--create a user-defined function that partially obfuscates credit card data
CREATE FUNCTION REDACT_CREDIT_CARD (credit_card TEXT)
RETURNS TEXT IMMUTABLE
AS $$
    import re
    regexp = re.compile("^([0-9]{6})[0-9]{5,6}([0-9]{4})")

    match = regexp.search(credit_card)
    if match != None:
        first = match.group(1)
        last = match.group(2)
    else:
        first = "000000"
        last = "0000"

    return "{}XXXXX{}".format(first, last)
$$ LANGUAGE plpythonu;

--create a masking policy that applies the REDACT_CREDIT_CARD function
CREATE MASKING POLICY mask_credit_card_partial
WITH (credit_card VARCHAR(256))
USING (REDACT_CREDIT_CARD(credit_card));

--confirm the masking policies using the associated system views
SELECT * FROM svv_masking_policy;

SELECT * FROM svv_attached_masking_policy;
```

附加遮罩政策

將遮罩政策附加至信用卡資料表。

```
--attach mask_credit_card_full to the credit card table as the default policy
```

```
--all users will see this masking policy unless a higher priority masking policy is
 attached to them or their role
ATTACH MASKING POLICY mask_credit_card_full
ON credit_cards(credit_card)
TO PUBLIC;

--attach mask_credit_card_partial to the analytics role
--users with the analytics role can see partial credit card information
ATTACH MASKING POLICY mask_credit_card_partial
ON credit_cards(credit_card)
TO ROLE analytics_role
PRIORITY 10;

--confirm the masking policies are applied to the table and role in the associated
 system view
SELECT * FROM svv_attached_masking_policy;

--confirm the full masking policy is in place for normal users by selecting from the
 credit card table as regular_user
SET SESSION AUTHORIZATION regular_user;

SELECT * FROM credit_cards;

--confirm the partial masking policy is in place for users with the analytics role by
 selecting from the credit card table as analytics_user
SET SESSION AUTHORIZATION analytics_user;

SELECT * FROM credit_cards;
```

修改遮罩政策

下一節說明如何修改動態資料遮罩政策。

```
--reset session authorization to the default
RESET SESSION AUTHORIZATION;

--alter the mask_credit_card_full policy
ALTER MASKING POLICY mask_credit_card_full
USING ('00000000000000'::TEXT);

--confirm the full masking policy is in place after altering the policy, and that
 results are altered from '000000XXXX0000' to '00000000000000'
SELECT * FROM credit_cards;
```

分離並捨棄遮罩政策

下列區段說明如何移除資料表中的所有動態資料遮罩政策，以分離和捨棄遮罩政策。

```
--reset session authorization to the default
RESET SESSION AUTHORIZATION;

--detach both masking policies from the credit_cards table
DETACH MASKING POLICY mask_credit_card_full
ON credit_cards(credit_card)
FROM PUBLIC;

DETACH MASKING POLICY mask_credit_card_partial
ON credit_cards(credit_card)
FROM ROLE analytics_role;

--drop both masking policies
DROP MASKING POLICY mask_credit_card_full;

DROP MASKING POLICY mask_credit_card_partial;
```

使用動態資料遮罩時的考量

使用動態資料遮罩時，請考量下列內容：

- 查詢從資料表建立的物件 (例如檢視) 時，使用者會根據自己的遮罩政策 (而非建立物件之使用者的政策) 來看到結果。例如，具有分析師角色的使用者查詢由 secadmin 建立的檢視時，就會看到附加到分析師角色的遮罩政策的結果。
- 為了防止 EXPLAIN 命令暴露敏感遮罩政策篩選條件，只有具有 SYS_EXPLAIN_DDM 許可的使用者才能看到 EXPLAIN 輸出中套用的遮罩政策。依預設，使用者沒有 SYS_EXPLAIN_DDM 許可。

  以下是授與角色權限的語法。

  ```
  GRANT EXPLAIN MASKING TO ROLE rolename
  ```

  如需 EXPLAIN 命令的詳細資訊，請參閱 EXPLAIN。
- 具有不同角色的使用者可以根據所使用的篩選條件或聯結條件看到不同的結果。例如，如果執行命令的使用者套用可混淆特定資料欄的遮罩政策，則使用該資料欄的值在資料表上執行 SELECT 命令將會失敗。
- DDM 政策必須在任何述詞操作或預測之前套用。遮罩政策可能包含下列項目：

- 低成本常數操作，例如將值轉換為 null
- 中等成本操作，例如 HMAC 雜湊
- 高成本操作，例如呼叫外部 Lambda 使用者定義函數

如此一來，我們建議您盡可能使用簡單遮罩運算式。

- 您可以針對具有資料列層級安全政策的角色使用 DDM 政策，但請注意，RLS 政策會在 DDM 之前套用。動態資料遮罩表達式無法讀取受 RLS 保護的資料列。如需 RLS 的詳細資訊，請參閱 資料列層級安全性。
- 使用 COPY 命令從 parquet 複製到受保護的目標資料表時，您應該在 COPY 陳述式中明確指定資料欄。如需使用 COPY 對應資料欄的詳細資訊，請參閱 欄映射選項。
- DDM 政策無法附加至下列關係：
  - 系統表格和目錄
  - 外部資料表
  - 資料共用資料表
  - 具體化視觀表
  - 跨資料庫關係
  - 暫時資料表
  - 相關查詢
- DDM 政策可以包含查閱資料表。查閱資料表可以存在於 USING 子句中。下列關係類型無法用作查閱資料表：
  - 系統表格和目錄
  - 外部資料表
  - 資料共用資料表
  - 檢視、具體化視觀表和近期繫結視觀表
  - 跨資料庫關係
  - 暫時資料表
  - 相關查詢

以下是將遮罩政策附加至查閱資料表的範例。

```
--Create a masking policy referencing a lookup table
CREATE MASKING POLICY lookup_mask_credit_card WITH (credit_card TEXT) USING (
  CASE
    WHEN
```

```
      credit_card IN (SELECT credit_card_lookup FROM credit_cards_lookup)
    THEN '000000XXXX0000'
    ELSE REDACT_CREDIT_CARD(credit_card)
    END
  );

--Provides access to the lookup table via a policy attached to a role
GRANT SELECT ON TABLE credit_cards_lookup TO MASKING POLICY lookup_mask_credit_card;
```

- 如果遮罩政策會產生與目標資料欄類型和大小不相容的輸出，則您無法附加該遮罩政策。例如，您無法將輸出 12 個字元長字串的遮罩政策附加至 VARCHAR(10) 資料欄。Amazon Redshift 支援以下例外情況：
  - 輸入類型為 INTN 的遮罩政策可以附加至大小為 INTM 的政策，只要 M < N 即可。例如，BIGINT (INT8) 輸入政策可附加至 smallint (INT4) 資料欄。
  - 輸入類型為 NUMERIC 或 DECIMAL 的遮罩政策一律可以附加至 FLOAT 資料欄。
- DDM 政策不能與資料共用搭配使用。如果資料共用的資料生產者將 DDM 政策附加至資料共用中的資料表，則使用者無法從嘗試查詢資料表的資料取用者中存取資料表。附加了 DDM 政策的資料表無法新增至資料共用。
- 如果下列任一組態選項的值不符合工作階段的預設值，您就無法查詢已附加 DDM 咒測的關係：
  - `enable_case_sensitive_super_attribute`
  - `enable_case_sensitive_identifier`
  - `downcase_delimited_identifier`

  如果您嘗試查詢已附加 DDM 政策的關係，並看到「受 DDM 保護的關係不支援工作階段層級組太，因為區分大小寫設定與其預設值不同」的訊息，請考慮重設工作階段的組態選項。
- 當您佈建的叢集或無伺服器命名空間具有任何動態資料遮罩政策時，一般使用者會無法使用下列命令：

  ```
  ALTER <current_user> SET enable_case_sensitive_super_attribute/
  enable_case_sensitive_identifier/downcase_delimited_identifier
  ```

  建立 DDM 政策時，建議您變更一般使用者的預設組態選項設定，以符合建立政策時的工作階段組態選項設定。超級使用者和擁有 ALTER USER 權限的使用者可以使用參數群組設定或 ALTER USER 命令來執行此操作。如需有關參數群組的詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 參數群組。如需 ALTER USER 命令的相關資訊，請參閱 ALTER USER。
- 使用 CREATE VIEW 命令的一般使用者，無法取代具有附加的 DDM 政策的視觀表和近期繫結視觀表。若要以 DDM 政策取代視觀表或 LBV，請先卸離任何附加的 DDM 政策、取代視觀表或 LBV，然

後重新附加政策。具有 sys:secadmin 權限的超級使用者和一般使用者都可以在具有 DDM 政策，但未卸離政策的視觀表或 LBV 上使用 CREATE VIEW。

- 具有附加 DDM 政策的視觀表，無法參考系統資料表和視觀表。近期繫結視觀表可以參考系統資料表和視觀表。
- 具有附加 DDM 政策的近期繫結視觀表，無法參考資料湖 (例如 JSON 文件) 中的巢狀資料。
- 如果任何視觀表都參考近期繫結視觀表，則近期繫結視觀表便無法附加 DDM 政策。
- 附加至近期繫結視觀表的 DDM 政策，會依資料欄名稱附加。在查詢時，Amazon Redshift 會驗證附加到近期繫結視觀表的所有遮罩政策是否已成功套用，而且近期繫結視觀表的輸出資料欄類型是否符合附加的遮罩政策的類型。如果驗證失敗，Amazon Redshift 會傳回查詢的錯誤訊息。

管理動態資料遮罩政策

您可以執行下列動作：

- 若要建立 DDM 政策，請使用 CREATE MASKING POLICY 命令。

  以下是使用 SHA-2 雜湊函數建立遮罩政策的範例。

```
CREATE MASKING POLICY hash_credit
WITH (credit_card varchar(256))
USING (sha2(credit_card + 'testSalt', 256));
```

- 若要變更現有的 DDM 政策，請使用 ALTER MASKING POLICY 命令。

  下列是變更現有遮罩政策的範例。

```
ALTER MASKING POLICY hash_credit
USING (sha2(credit_card + 'otherTestSalt', 256));
```

- 若要將資料表上的 DDM 政策附加至一或多個使用者或角色，請使用 ATTACH MASKING POLICY 命令。

  下列是將遮罩政策附加至資料欄/角色組的範例。

```
 ATTACH MASKING POLICY hash_credit
ON credit_cards (credit_card)
TO ROLE science_role
PRIORITY 30;
```

PRIORITY 子句會決定當多個政策附加至相同資料欄時，哪個遮罩政策會套用至使用者工作階段。例如，如果上述範例中的使用者在相同的信用卡資料欄中附加了另一個遮罩政策，且優先順序為 20，那麼 science_role 的政策就是適用的政策，因為其具有較高的優先順序 30。

- 若要將資料表上的 DDM 政策與一或多個使用者或角色分離，請使用 DETACH MASKING POLICY 命令。

  下列是將遮罩政策從資料欄/角色組分離的範例。

```
DETACH MASKING POLICY hash_credit
ON credit_cards(credit_card)
FROM ROLE science_role;
```

- 若要從所有資料庫捨棄 DDM 政策，請使用 DROP MASKING POLICY 命令。

  以下是從所有資料庫捨棄遮罩政策的範例。

```
DROP MASKING POLICY hash_credit;
```

遮罩政策階層

附加多項遮罩政策時，請考慮下列事項：

- 您可以將多個遮罩政策附加至單一資料欄。
- 當查詢適用多個遮罩政策時，則會套用附加至每個資料欄且優先順序最高的政策。請考量下列範例。

```
ATTACH MASKING POLICY partial_hash
ON credit_cards(address, credit_card)
TO ROLE analytics_role
PRIORITY 20;

ATTACH MASKING POLICY full_hash
ON credit_cards(credit_card, ssn)
TO ROLE auditor_role
PRIORITY 30;

SELECT address, credit_card, ssn
FROM credit_cards;
```

執行 SELECT 陳述式時，具有分析和稽核角色的使用者會看到已套用 partial_hash 遮罩政策的位址欄。他們會看到套用 full_hash 遮罩政策的信用卡和 SSN 資料欄，因為原 full_hash 政策在信用卡資料欄上具有較高的優先權。

- 如果附加遮罩政策時未指定優先順序，則預設的優先順序為 0。
- 您無法將兩個政策附加至具有相同優先順序的相同資料欄。
- 您無法將兩個政策附加到使用者和欄或角色和欄的相同組合。
- 當連結至相同使用者或角色時，沿著相同的 SUPER 路徑套用多個遮罩政策時，只有優先順序最高的附件才會生效。請考慮下列範例。

  第一個範例顯示在相同路徑上附加的兩個遮罩政策，優先順序較高的政策會生效。

```
ATTACH MASKING POLICY hide_name
ON employees(col_person.name)
TO PUBLIC
PRIORITY 20;

ATTACH MASKING POLICY hide_last_name
ON employees(col_person.name.last)
TO PUBLIC
PRIORITY 30;

--Only the hide_last_name policy takes effect.
SELECT employees.col_person.name FROM employees;
```

  第二個範例顯示連接至相同 SUPER 物件中不同路徑的兩個遮罩政策，因此在政策之間不會發生衝突。兩項附件同時採用。

```
ATTACH MASKING POLICY hide_first_name
ON employees(col_person.name.first)
TO PUBLIC
PRIORITY 20;

ATTACH MASKING POLICY hide_last_name
ON employees(col_person.name.last)
TO PUBLIC
PRIORITY 20;

--Both col_person.name.first and col_person.name.last are masked.
SELECT employees.col_person.name FROM employees;
```

若要確認哪個遮罩政策適用於指定的使用者和資料欄，或角色和資料欄的組合，具有 sys:secadmin 角色的使用者可以在 SVV_ATTACHED_MASKING_POLICY 系統檢視中查詢資料欄/角色或資料欄/使用者配對。如需詳細資訊，請參閱 動態資料遮罩的系統檢視。

搭配 SUPER 資料類型路徑使用動態資料遮罩

Amazon Redshift 支援將動態資料遮罩政策附加到 SUPER 類型資料欄的路徑。如需 SUPER 資料類型的相關資訊，請參閱 在 Amazon Redshift 中擷取和查詢半結構化資料。

將遮罩政策附加至 SUPER 類型資料欄的路徑時，請考慮下列事項。

- 將遮罩政策附加至資料欄上的路徑時，必須將該資料欄定義為 SUPER 資料類型。您只能將遮罩政策套用至 SUPER 路徑上的純量值。您無法將遮罩政策套用至複雜的結構或陣列。
- 只要 SUPER 路徑不衝突，您就可以將不同的遮罩政策套用至單一 SUPER 資料欄上的多個純量值。例如，SUPER 路徑 `a.b` 和 `a.b.c` 衝突，因為它們位於相同路徑，而且 `a.b` 是 `a.b.c` 的上層。SUPER 路徑`a.b.c` 和 `a.b.d` 沒有衝突。
- 在使用者查詢執行期套用政策之前，Amazon Redshift 無法檢查遮罩政策附加的路徑是否存在於資料中，且為預期類型的路徑。例如，當您將遮罩 TEXT 值的遮罩政策附加到包含 INT 值的 SUPER 路徑時，Amazon Redshift 會嘗試在路徑上轉換為該值的類型。

  在這種情況下，Amazon Redshift 在執行期的行為取決於您用於查詢 SUPER 物件的組態設定。根據預設，Amazon Redshift NULL處於寬鬆模式，並且會像指定的 SUPER 路徑一樣解析遺失的路徑和無效轉換。如需與 SUPER 相關之組態設定的詳細資訊，請參閱 SUPER 組態。
- SUPER 是一種無結構描述類型，這表示 Amazon Redshift 無法確認在指定的 SUPER 路徑上的值是否存在。如果您將遮罩政策附加到不存在的 SUPER 路徑，且 Amazon Redshift 處於寬鬆模式，則 Amazon Redshift 會將路徑解析為 NULL 值。建議您在將遮罩政策附加至 SUPER 資料欄的路徑時，考慮 SUPER 物件的預期格式，以及具有未預期屬性的可能性。如果您認為 SUPER 資料欄中可能有未預期的結構描述，請考慮將遮罩政策直接附加至 SUPER 資料欄。您可以使用 SUPER 類型資訊函數來檢查屬性和類型，並使用 `OBJECT_TRANSFORM` 遮罩值。如需 SUPER 類型資訊函數的詳細資訊，請參閱 SUPER 類型資訊函數。

範例

將遮罩政策附加至 SUPER 路徑

下列範例會將多個遮罩政策附加至一個資料欄中的多個 SUPER 類型路徑。

```
CREATE TABLE employees (
```

```
    col_person SUPER
);

INSERT INTO employees
VALUES
    (
        json_parse('
            {
                "name": {
                    "first": "John",
                    "last": "Doe"
                },
                "age": 25,
                "ssn": "111-22-3333",
                "company": "Company Inc."
            }
        ')
    ),
    (
        json_parse('
            {
                "name": {
                    "first": "Jane",
                    "last": "Appleseed"
                },
                "age": 34,
                "ssn": "444-55-7777",
                "company": "Organization Org."
            }
        ')
    )
;
GRANT ALL ON ALL TABLES IN SCHEMA "public" TO PUBLIC;

-- Create the masking policies.

-- This policy converts the given name to all uppercase letters.
CREATE MASKING POLICY mask_first_name
WITH(first_name TEXT)
USING ( UPPER(first_name) );

-- This policy replaces the given name with the fixed string 'XXXX'.
CREATE MASKING POLICY mask_last_name
WITH(last_name TEXT)
```

```
USING ( 'XXXX'::TEXT );

-- This policy rounds down the given age to the nearest 10.
CREATE MASKING POLICY mask_age
WITH(age INT)
USING ( (FLOOR(age::FLOAT / 10) * 10)::INT );

-- This policy converts the first five digits of the given SSN to 'XXX-XX'.
CREATE MASKING POLICY mask_ssn
WITH(ssn TEXT)
USING ( 'XXX-XX-'::TEXT || SUBSTRING(ssn::TEXT FROM 8 FOR 4) );

-- Attach the masking policies to the employees table.
ATTACH MASKING POLICY mask_first_name
ON employees(col_person.name.first)
TO PUBLIC;

ATTACH MASKING POLICY mask_last_name
ON employees(col_person.name.last)
TO PUBLIC;

ATTACH MASKING POLICY mask_age
ON employees(col_person.age)
TO PUBLIC;

ATTACH MASKING POLICY mask_ssn
ON employees(col_person.ssn)
TO PUBLIC;

-- Verify that your masking policies are attached.
SELECT
    policy_name,
    TABLE_NAME,
    priority,
    input_columns,
    output_columns
FROM
    svv_attached_masking_policy;

   policy_name   | table_name | priority |           input_columns           |
 output_columns
-----------------+------------+----------+-----------------------------------
+-----------------------------------
```

```
 mask_age        | employees  |        0 | ["col_person.\"age\""]            |
 ["col_person.\"age\""]
 mask_first_name | employees  |        0 | ["col_person.\"name\".\"first\""] |
 ["col_person.\"name\".\"first\""]
 mask_last_name  | employees  |        0 | ["col_person.\"name\".\"last\""]  |
 ["col_person.\"name\".\"last\""]
 mask_ssn        | employees  |        0 | ["col_person.\"ssn\""]            |
 ["col_person.\"ssn\""]
(4 rows)

-- Observe the masking policies taking effect.
SELECT col_person FROM employees ORDER BY col_person.age;

-- This result is formatted for ease of reading.
         col_person
--------------------------------
{
    "name": {
        "first": "JOHN",
        "last": "XXXX"
    },
    "age": 20,
    "ssn": "XXX-XX-3333",
    "company": "Company Inc."
}
{
    "name": {
        "first": "JANE",
        "last": "XXXX"
    },
    "age": 30,
    "ssn": "XXX-XX-7777",
    "company": "Organization Org."
}
```

以下是 SUPER 路徑的無效遮罩政策附件的一些範例。

```
-- This attachment fails because there is already a policy
-- with equal priority attached to employees.name.last, which is
-- on the same SUPER path as employees.name.
ATTACH MASKING POLICY mask_ssn
ON employees(col_person.name)
TO PUBLIC;
```

```
ERROR:  DDM policy "mask_last_name" is already attached on relation "employees" column
 "col_person."name"."last"" with same priority

-- Create a masking policy that masks DATETIME objects.
CREATE MASKING POLICY mask_date
WITH(INPUT DATETIME)
USING ( INPUT );

-- This attachment fails because SUPER type columns can't contain DATETIME objects.
ATTACH MASKING POLICY mask_date
ON employees(col_person.company)
TO PUBLIC;
ERROR:  cannot attach masking policy for output of type "timestamp without time zone"
 to column "col_person."company"" of type "super
```

以下是將遮罩政策附加至不存在的 SUPER 路徑的範例。預設情況下，Amazon Redshift 會將路徑解析到 NULL。

```
ATTACH MASKING POLICY mask_first_name
ON employees(col_person.not_exists)
TO PUBLIC;

SELECT col_person FROM employees LIMIT 1;

-- This result is formatted for ease of reading.
         col_person
-----------------------------------
{
    "name": {
        "first": "JOHN",
        "last": "XXXX"
    },
    "age": 20,
    "ssn": "XXX-XX-3333",
    "company": "Company Inc.",
    "not_exists": null
}
```

條件式動態資料遮罩

您可以在遮罩運算式中使用條件運算式建立遮罩政策，以在儲存格層級上遮罩資料。例如，您可以建立一個遮罩政策，根據該資料列中其他資料欄的值，將不同的遮罩套用至值。

以下是使用條件式資料遮罩來建立並附加遮罩政策的範例，該遮罩政策會修改涉及詐騙的部分信用卡號碼，同時完全隱藏所有其他信用卡號碼。您必須是超級使用者或具有 sys:secadmin 角色才能執行此範例。

```
--Create an analyst role.
CREATE ROLE analyst;

--Create a credit card table. The table contains an is_fraud boolean column,
--which is TRUE if the credit card number in that row was involved in a fraudulent
 transaction.
CREATE TABLE credit_cards (id INT, is_fraud BOOLEAN, credit_card_number VARCHAR(16));

--Create a function that partially redacts credit card numbers.
CREATE FUNCTION REDACT_CREDIT_CARD (credit_card VARCHAR(16))
RETURNS VARCHAR(16) IMMUTABLE
AS $$
    import re
    regexp = re.compile("^([0-9]{6})[0-9]{5,6}([0-9]{4})")

    match = regexp.search(credit_card)
    if match != None:
        first = match.group(1)
        last = match.group(2)
    else:
        first = "000000"
        last = "0000"

    return "{}XXXXX{}".format(first, last)
$$ LANGUAGE plpythonu;

--Create a masking policy that partially redacts credit card numbers if the is_fraud
 value for that row is TRUE,
--and otherwise blanks out the credit card number completely.
CREATE MASKING POLICY card_number_conditional_mask
    WITH (fraudulent BOOLEAN, pan varchar(16))
    USING (CASE WHEN fraudulent THEN REDACT_CREDIT_CARD(pan)
                ELSE Null
           END);

--Attach the masking policy to the credit_cards/analyst table/role pair.
ATTACH MASKING POLICY card_number_conditional_mask ON credit_cards (credit_card_number)
 USING (is_fraud, credit_card_number)
 TO ROLE analyst PRIORITY 100;
```

動態資料遮罩的系統檢視

超級使用者、具有該sys:operator角色的使用者以及具有 ACCESS SYSTEM 表格權限的使用者可以存取下列與 DDM 相關的系統檢視。

- SVV_MASKING_POLICY

  使用 SVV_MASKING_POLICY 可檢視在叢集或工作群組上建立的所有遮罩原則。
- SVV_ATTACHED_MASKING_POLICY

  您可以使用 SVV_ATTACHED_MASKING_POLICY 來檢視目前連線資料庫上附加原則的所有關係和使用者或角色。
- 系統應用程序掩碼策略日誌

  使用 SYS_APPLICKING_LOG 來追蹤參考受 DDM 保護之關係之查詢的遮罩原則應用程式。

以下是您可以使用系統檢視找到的一些資訊範例。

```
--Select all policies associated with specific users, as opposed to roles
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE grantee_type = 'user';

--Select all policies attached to a specific user
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE grantee = 'target_grantee_name'

--Select all policies attached to a given table
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE table_name = 'target_table_name'
```

```
      AND schema_name = 'target_schema_name';

--Select the highest priority policy attachment for a given role
SELECT samp.policy_name,
       samp.priority,
       samp.grantee,
       smp.policy_expression
FROM svv_masking_policy AS smp
JOIN svv_attached_masking_policy AS samp
    ON samp.policy_name = smp.policy_name
WHERE
    samp.grantee_type = 'role' AND
    samp.policy_name = mask_get_policy_for_role_on_column(
        'target_schema_name',
        'target_table_name',
        'target_column_name',
        'target_role_name')
ORDER BY samp.priority desc
LIMIT 1;

--See which policy a specific user will see on a specific column in a given relation
SELECT samp.policy_name,
       samp.priority,
       samp.grantee,
       smp.policy_expression
FROM svv_masking_policy AS smp
JOIN svv_attached_masking_policy AS samp
    ON samp.policy_name = smp.policy_name
WHERE
    samp.grantee_type = 'role' AND
    samp.policy_name = mask_get_policy_for_user_on_column(
        'target_schema_name',
        'target_table_name',
        'target_column_name',
        'target_user_name')
ORDER BY samp.priority desc;

 --Select all policies attached to a given relation.
SELECT policy_name,
schema_name,
relation_name,
database_name
FROM sys_applied_masking_policy_log
WHERE relation_name = 'relation_name'
```

```
AND schema_name = 'schema_name';
```

限定範圍權限

作用域權限可讓您將權限授與資料庫或結構描述中某個類型之所有物件的使用者或角色。具有範圍權限的使用者和角色對資料庫或結構描述中所有目前和 future 物件都具有指定的權限。

如需套用限定範圍的權限的詳細資訊，請參閱 GRANT 和 REVOKE。

使用限定範圍的權限考量

使用限定範圍的權限時，請考量下列事項：

- 您可以使用範圍的權限，將資料庫或結構描述範圍的權限授與或撤銷指定使用者或角色之間的權限。
- 您無法將範圍權限授與使用者群組。
- 授予或撤銷限定範圍的權限會變更範圍中所有目前和未來物件的權限。
- 設定範圍的權限和物件層級權限彼此獨立運作。例如，在下列兩種情況下，使用者將維護資料表的權限。
  - 在資料表結構描述 1.table1 上授與使用者 SELECT，並在結構描述 1 上獲得選取範圍的權限。然後，使用者已撤銷綱要結構描述 1 中所有資料表的 SELECT。使用者會在模式 1 表格 1 上保留 SELECT。
  - 在資料表結構描述 1.table1 上授與使用者 SELECT，並在結構描述 1 上獲得選取範圍的權限。然後，用戶已撤銷模式 1 的 SELECT。使用者會在模式 1 表格 1 上保留 SELECT。
- 若要授予或撤銷限定範圍的權限，您必須符合下列其中一項條件：
  - 超級使用者。
  - 具有該權限授予選項的使用者。如需有關授權選項的詳細資訊，請移至中的「具有授權選項」參數GRANT。
- 限定範圍的權限只能授予或撤銷已連線資料庫的物件，或從資料共用匯入的資料庫撤銷。
- 您可以使用範圍的權限來設定從資料清單建立之資料庫的預設權限。被授予共用資料庫限定範圍權限的取用者端資料共用使用者，將自動獲得新增到生產者端資料共用之任何新物件的那些權限。
- 生產者可以將結構描述內物件的範圍權限授與資料清單。 (預覽)

SQL 參考

主題

Amazon Redshift SQL

主題

Amazon Redshift 是以業界標準 SQL 為基礎所建置的，並新增功能，用來管理超大資料集，和支援對這些資料的高效能分析與報告。

> **Note**
>
> 單一 Amazon Redshift SQL 陳述式的大小上限為 16 MB。

領導節點上所支援的 SQL 函數

某些 Amazon Redshift 查詢會分配到運算節點上執行，其他的查詢則是只在領導節點上執行。

每當查詢參考使用者建立的資料表或系統資料表 (具有 STL 或 STV 字首的資料表，以及具有 SVL 或 SVV 字首的系統畫面) 時，領導節點就會將 SQL 分送到運算節點。查詢如果只參考目錄資料表 (具有 PG 字首的資料表，例如 PG_TABLE_DEF，儲存於領導節點上)，或是未參考任何資料表，就只會在領導節點上執行。

某些 Amazon Redshift SQL 函數只有在領導者節點上才支援，在運算節點上不支援。使用領導者節點函數的查詢必須完全在領導者節點上執行，而不是在運算節點上，否則會傳回錯誤。

必須只在領導節點上執行的每個函數，其文件包含備註，說明如果函數參考使用者定義的資料表或 Amazon Redshift 系統資料表，將會傳回錯誤。關於只在領導節點上執行的功能，如需功能清單，請參閱 僅限領導節點函數。

範例

下列範例使用範例 TICKIT 資料庫。如需範例資料庫的詳細資訊，請移至範本資料庫。

CURRENT_SCHEMA

CURRENT_SCHEMA 是只限領導節點的函式。在此範例中，查詢並未參考資料表，因此只會在領導節點上執行。

```
select current_schema();

current_schema
---------------
public
```

在下一個範例中，查詢參考了系統目錄資料表，因此只會在領導節點上執行。

```
select * from pg_table_def
where schemaname = current_schema() limit 1;

 schemaname | tablename | column |   type   | encoding | distkey | sortkey | notnull
------------+-----------+--------+----------+----------+---------+---------+---------
 public     | category  | catid  | smallint | none     | t       |       1 | t
```

在下一個範例中，查詢參考了位於運算節點上的 Amazon Redshift 系統資料表，因此會傳回錯誤。

```
select current_schema(), userid from users;

INFO:  Function "current_schema()" not supported.
ERROR:  Specified types or functions (one per INFO message) not supported on Amazon
 Redshift tables.
```

SUBSTR

SUBSTR 也是一個僅限於領導節點函數。在以下範例中，因查詢未參考資料表，所以只會在領導節點上執行。

```
SELECT SUBSTR('amazon', 5);
```

```
+--------+
| substr |
+--------+
| on     |
+--------+
```

在下列範例中，查詢會參照位於運算節點上的資料表。這會導致錯誤。

```
SELECT SUBSTR(catdesc, 1) FROM category LIMIT 1;

ERROR: SUBSTR() function is not supported (Hint: use SUBSTRING instead)
```

若要成功執行先前的查詢，請使用 SUBSTRING。

```
SELECT SUBSTRING(catdesc, 1) FROM category LIMIT 1;

+---------------------------------+
|            substring            |
+---------------------------------+
| National Basketball Association |
+---------------------------------+
```

階乘 ( )

階乘 ( ) 是一個領導者節點的唯一函數。在以下範例中，因查詢未參考資料表，所以只會在領導節點上執行。

```
SELECT FACTORIAL(5);

 factorial
-------------
 120
```

在下列範例中，查詢會參照位於運算節點上的資料表。使用查詢編輯器 v2 執行時，會導致錯誤。

```
create table t(a int);
insert into t values (5);
select factorial(a) from t;

ERROR: Specified types or functions (one per INFO message) not supported on Redshift
 tables.
```

```
Info: Function "factorial(bigint)" not supported.
```

Amazon Redshift 和 PostgreSQL

主題

Amazon Redshift 是以 PostgreSQL 為基礎。在設計和開發您的資料倉儲應用程式時，您必須知道 Amazon Redshift 與 PostgreSQL 之間有多項重要的差異。

Amazon Redshift 是專門針對線上分析處理 (OLAP) 和商業智慧 (BI) 應用所設計的，這些應用需要對龐大的資料集進行複雜的查詢。由於 Amazon Redshift 是用來滿足極為不同的需求，因此所使用的專業資料儲存體架構和查詢執行引擎，和 PostgreSQL 實作完全不同。例如，線上交易處理 (OLTP) 應用程式通常將資料儲存在列中，而 Amazon Redshift 則把資料儲存在欄中，並使用專門的資料壓縮編碼來實現最佳記憶體使用和磁碟 I/O。為了提高效能，一些適合較小規模 OLTP 處理的 PostgreSQL 功能 (例如次要索引和高效的單列資料操作操作) 已被省略。

如需 Amazon Redshift 資料倉儲系統架構的詳細說明，請參閱系統和架構概觀。

PostgreSQL 9.x 包含了一些 Amazon Redshift 未支援的功能。此外，Amazon Redshift SQL 和 PostgreSQL 之間有幾項重要的差異，是您必須知道的。本節重點介紹了 Amazon Redshift 和 PostgreSQL 之間的差異，並提供開發資料倉儲的指導方針，以充分善用 Amazon Redshift SQL 實作。

Amazon Redshift 和 PostgreSQL JDBC 和 ODBC

由於 Amazon Redshift 是以 PostgreSQL 為基礎，因此我們先前建議使用 JDBC4 Postgresql 驅動程式版本 8.4.703 和 psqlODBC 版本 9.x 驅動程式。如果您目前正在使用這些驅動程式，我們建議今後移至新的 Amazon Redshift 專屬驅動程式。如需驅動程式和設定連線的相關資訊，請參閱《Amazon Redshift 管理指南》中的適用於 Amazon Redshift 的 JDBC 和 ODBC 驅動程式。

若要避免使用 JDBC 擷取大型資料集時發生用戶端 out-of-memory 錯誤，您可以設定 JDBC 擷取大小參數，讓用戶端能夠批次擷取資料。如需詳細資訊，請參閱 設定 JDBC 擷取大小參數。

Amazon Redshift 無法辨識 JDBC maxRows 參數。請改為指定 LIMIT 子句來限制結果集。您也可以使用 OFFSET 子句，來跳到結果集中的指定起點。

以不同方式建置的功能

許多的 Amazon Redshift SQL 語言元素具有不同的效能特性，而且所使用的語法和語意，也和同等 PostgreSQL 實作非常不同。

> **Important**
>
> 請勿假設 Amazon Redshift 和 PostgreSQL 共通的元素具有相同的語意。請務必參考《Amazon Redshift 開發人員指南》的 SQL 命令，以了解細微的差異。

一個特別的例子是 VACUUM 命令，這是用來清理和重整資料表。VACUUM 具有不同的功能，而且使用與 PostgreSQL 版本不同的一組參數。如需在 Amazon Redshift 中使用 VACUUM 的詳細資訊，請參閱清空資料表。

資料庫管理的功能和工具也經常有所不同。例如，Amazon Redshift 會維持一組系統資料表和檢視，這些項目提供關於系統運作狀況的資訊。如需詳細資訊，請參閱系統資料表和檢視。

下列清單包含了一些 SQL 功能的範例，這些功能在 Amazon Redshift 中以不同的方式建置。

- CREATE TABLE

  Amazon Redshift 不支援資料表空間、資料表分割、繼承和某些限制。CREATE TABLE 的 Amazon Redshift 實作可讓您定義資料表的排序與分佈演算法，以實現最佳化的平行處理。

  Amazon Redshift Spectrum 支援使用 CREATE EXTERNAL TABLE 命令來分割資料表。
- ALTER TABLE

  僅支援 ALTER COLUMN 動作的子集。

  ADD COLUMN 支援在每個 ALTER TABLE 陳述式中只新增一欄。
- COPY

  Amazon Redshift COPY 是高度專門的命令，可用來從 Amazon S3 儲存貯體和 Amazon DynamoDB 資料表載入資料，以便於自動壓縮作業的進行。如需詳細資訊，請參閱 載入資料 區段和 COPY 指令參考。
- VACUUM

VACUUM 所使用的參數完全不同。例如，PostgreSQL 中的預設 VACUUM 操作，只會回收空間，以供重複使用；但是，Amazon Redshift 中預設的 VACUUM 操作為 VACUUM FULL，這項操作會回收磁碟空間，並將所有列重新排序。

- 在比較字串值時，會忽略 VARCHAR 值中多餘的空格。如需詳細資訊，請參閱 多餘空格的意義。

不支援的 PostgreSQL 功能

在 Amazon Redshift 中不支援下列的 PostgreSQL 功能。

> **Important**
>
> 請勿假設 Amazon Redshift 和 PostgreSQL 共通的元素具有相同的語意。請務必參考《Amazon Redshift 開發人員指南》的 SQL 命令，以了解細微的差異。

- 不支援查詢工具 psql。支援 Amazon Redshift RSQL 用戶端。
- 資料表分割 (範圍與列表分割)
- 資料表空間
- 限制
  - 唯一
  - 外部索引鍵
  - 主索引鍵
  - 檢查限制
  - 排除限制

  允許唯一、主索引鍵和外部索引鍵的限制，但這些只是參考資訊。系統不會強制執行這些限制，不過查詢規劃器會使用這些限制。
- 資料庫角色
- 繼承
- PostgreSQL 系統欄

  Amazon Redshift SQL 不會隱含地定義系統欄。但是，下列 PostgreSQL 系統欄的名稱不能做為使用者定義欄的名稱使用：`oid`、`tableoid`、`xmin`、`cmin`、`xmax`、`cmax` 和 `ctid`。如需詳細資訊，請參閱 https://www.postgresql.org/docs/8.0/static/ddl-system-columns.html。

- 索引
- Window 函式中的 NULLS 子句
- 定序

  Amazon Redshift 不支援地區特定或使用者定義的定序句。請參閱定序序列。
- 值表達式
  - 註標表達式
  - 陣列建構子
  - 列建構子
- 觸發
- 外部資料管理 (SQL/MED)
- 資料表函式
- 做為常數表使用的 VALUES 清單
- 序列
- 全文檢索搜尋

不支援的 PostgreSQL 資料類型

一般而言，如果查詢嘗試使用不支援的資料類型 (包括明確或隱含轉換)，將會傳回錯誤。不過，使用不支援資料類型的某些查詢，將可在領導節點上執行，但無法在運算節點上執行。請參閱領導節點上所支援的 SQL 函數。

如需支援的資料類型清單，請參閱 資料類型。

在 Amazon Redshift 中不支援下列的 PostgreSQL 資料類型。

- 陣列
- BIT、BIT VARYING
- BYTEA
- 複合類型
- 日期/時間類型
- 列舉類型
- 幾何類型

- HSTORE
- JSON
- 網路位址類型
- 數值類型
  - SERIAL、BIGSERIAL、SMALLSERIAL
  - MONEY
- 物件識別碼類型
- 虛擬類型
- 範圍類型
- 特殊字元類型
  - "char" - 單位元組的內部類型 (其中名為 char 的資料類型會用引號括住)。
  - name - 物件名稱的內部類型。

  如需這些類型的相關資訊，請參閱 PostgreSQL 文件中的 Special Character Types。
- 文字搜尋類型
- TXID_SNAPSHOT
- UUID
- XML

不支援的 PostgreSQL 函數

許多未排除的函式具有不同的語意或用法。例如，某些支援的函式只會在領導節點上執行。此外，某些不支援的函式在領導節點上執行時，不會傳回錯誤。這些函數在某些情況中不會傳回錯誤的事實，不應視為 Amazon Redshift 支援這些函數。

> **Important**
>
> 請勿假設 Amazon Redshift 和 PostgreSQL 共通的元素具有相同的語意。請務必參考《Amazon Redshift 資料庫開發人員指南》的 SQL 命令，以了解細微的差異。

如需詳細資訊，請參閱 領導節點上所支援的 SQL 函數。

在 Amazon Redshift 中不支援下列的 PostgreSQL 函數。

- 存取權限查詢函式
- 建議鎖函式
- 彙總函數
  - STRING_AGG()
  - ARRAY_AGG()
  - EVERY()
  - XML_AGG()
  - CORR()
  - COVAR_POP()
  - COVAR_SAMP()
  - REGR_AVGX()、REGR_AVGY()
  - REGR_COUNT()
  - REGR_INTERCEPT()
  - REGR_R2()
  - REGR_SLOPE()
  - REGR_SXX()、REGR_SXY()、REGR_SYY()
- 陣列函數和運算子
- 備份控制函式
- 註解資訊函式
- 資料庫物件位置函式
- 資料庫物件大小函式
- 日期/時間函式與運算子
  - CLOCK_TIMESTAMP()
  - JUSTIFY_DAYS()、JUSTIFY_HOURS()、JUSTIFY_INTERVAL()
  - PG_SLEEP()
  - TRANSACTION_TIMESTAMP()
- ENUM 支援函式
- 幾何函式與運算子
- 通用檔案存取函式
- IS DISTINCT FROM

- 網路位址函式與運算子
- 數學函式
  - DIV()
  - SETSEED()
  - WIDTH_BUCKET()
- 結果集傳回函式
  - GENERATE_SERIES()
  - GENERATE_SUBSCRIPTS()
- 範圍函式與運算子
- 復原控制函式
- 復原資訊函式
- ROLLBACK TO SAVEPOINT 函式
- 結構描述可見性查詢函式
- 伺服器發訊函式
- 快照同步函式
- 序列處理函式
- 字串函數
  - BIT_LENGTH()
  - OVERLAY()
  - CONVERT()、CONVERT_FROM()、CONVERT_TO()
  - ENCODE()
  - FORMAT()
  - QUOTE_NULLABLE()
  - REGEXP_MATCHES()
  - REGEXP_SPLIT_TO_ARRAY()
  - REGEXP_SPLIT_TO_TABLE()
- 系統目錄資訊函式
- 系統資訊函數
  - CURRENT_CATALOG CURRENT_QUERY()
  - INET_CLIENT_ADDR()

  - INET_CLIENT_PORT()
  - INET_SERVER_ADDR() INET_SERVER_PORT()
  - PG_CONF_LOAD_TIME()
  - PG_IS_OTHER_TEMP_SCHEMA()
  - PG_LISTENING_CHANNELS()
  - PG_MY_TEMP_SCHEMA()
  - PG_POSTMASTER_START_TIME()
  - PG_TRIGGER_DEPTH()
  - SHOW VERSION()
- 文字搜尋函式與運算子
- 異動 ID 與快照函式
- 觸發函數
- XML 函式

使用 SQL

主題

- SQL 參考慣例
- 基本元素
- 表達式
- 條件

SQL 語言包含指令與函式，可用來操作資料庫與資料庫物件。此語言也會針對資料類型、表達式和常值的使用，來強制執行規則。

SQL 參考慣例

本節針對在 SQL 參考章節中所說明的 SQL 表達式、指令與函式，說明撰寫其語法時所使用的慣例。

字元	描述
CAPS (大寫字母)	大寫字詞為關鍵字。

字元	描述
[ ]	方括號是用來表示選用的引數。方括號中的多個引數，代表您可以選擇任何數目的引數。此外，方括號中不同行的引數，代表 Amazon Redshift 剖析器期望引數的排序，等於在語法中所列出的順序。如需範例，請參閱SELECT。
{ }	大括號表示您需要選擇大括號內的其中一個引數。
|	直立線符號會將您可選擇的引數隔開。
斜體	斜體的字表示預留位置。您必須插入適當的值來取代斜體字詞。
. . .	省略符號表示您可以重複前一個元素。
'	單引號中的字詞表示您必須輸入引用內容。

基本元素

主題

- 名稱與識別碼
- 文字
- Null
- 資料類型
- 定序序列

本節針對資料庫物件名稱、常值、null 及資料類型，說明了使用的規則。

名稱與識別碼

名稱可用來識別資料庫物件，包括資料表和資料欄，以及使用者和密碼。名稱和識別碼這兩個詞可互換使用。識別碼有兩種：標準識別碼和引號 (或區隔) 識別碼。識別碼必須只能包含 UTF-8 可列印字元。標準識別碼和區隔識別碼中的 ASCII 字母，應區分大小寫，而且在資料庫中轉換為小寫。在查詢結果中，欄的名稱預設會以小寫傳回。若要以大寫傳回欄的名稱，請將 describe_field_name_in_uppercase 組態參數設定為 **`true`**。

標準識別碼

標準 SQL 識別碼需遵守一套規則，而且必須：

- 開頭為 ASCII 單位元組字母字元、底線字元或 UTF-8 多位元組字元 (長度 2 到 4 位元組)。
- 後續的字元可以是 ASCII 單位元組字母字元、底線字元、金額符號或 UTF-8 多位元組字元 (長度 2 到 4 位元組)。
- 讓長度介於 1 到 127 個位元組之間，不包括區隔識別碼的引號。
- 不包含引號和空格。
- 不能是保留的 SQL 關鍵字。

區隔識別碼

區隔識別碼 (也稱為引號識別碼) 是以雙引號 (") 開頭和結尾。如果使用區隔識別碼，則每次參考該物件，都必須使用雙引號。除了雙引號本身以外，識別碼可包含任何標準 UTF-8 可列印字元。因此，您可以建立欄或資料表的名稱，其中包含無效字元，例如空格或百分比符號。

區隔識別碼中的 ASCII 字母，應區分大小寫，並且轉換為小寫。若要在字串中使用雙引號，必須在其前面加上另一個雙引號字元。

區分大小寫的識別碼

區分大小寫的識別碼 (也稱為混合大小寫識別碼) 可以同時包含大寫和小寫字母。若要使用區分大小寫的識別碼，您可以將組態 `enable_case_sensitive_identifier` 設定為 `true`。您可以為叢集或工作階段設定此組態。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的預設參數值和enable_case_sensitive_identifier。

系統欄名稱

下列 PostgreSQL 系統欄名稱不能做為使用者定義欄的欄名稱使用。如需詳細資訊，請參閱 https://www.postgresql.org/docs/8.0/static/ddl-system-columns.html。

- `oid`
- `tableoid`
- `xmin`
- `cmin`
- `xmax`
- `cmax`
- `ctid`

範例

下表顯示區隔識別碼的範例、產生的輸出和說明：

語法	結果	說明
"group"	群組	GROUP 是一個保留字詞，因此如果在識別碼中使用，需要加上雙引號。
"""WHERE"""	"where"	WHERE 也是一個保留字詞。若要在字串中包含雙引號，請使用額外的雙引號字元，來逸出每個雙引號字元。
"This name"	this name	必須使用雙引號來保留空格。
"This ""IS IT"""	this "is it"	包夾 IS IT 的引號，前後必須再加上另一個引號，才能成為名稱的一部分。

若要建立名為 group 的資料表，而且此資料表包含名為 this "is it" 的欄：

```
create table "group" (
"This ""IS IT""" char(10));
```

下列的查詢會傳回相同的結果：

```
select "This ""IS IT"""
from "group";

this "is it"
--------------
(0 rows)
```

```
select "this ""is it"""
from "group";

this "is it"
--------------
(0 rows)
```

下列的完整 `table.column` 語法，也會傳回相同的結果：

```
select "group"."this ""is it"""
from "group";

this "is it"
--------------
(0 rows)
```

下列的 CREATE TABLE 命令會建立資料欄名稱中有一條斜線的資料表：

```
create table if not exists city_slash_id(
                  "city/id" integer not null,
                  state char(2) not null);
```

文字

常值或常數是固定的資料值，由一系列的字元或數值常數構成。Amazon Redshift 支援多種類型的常值，包括：

- 整數的數值常值、小數和符點數。如需詳細資訊，請參閱 整數常值與浮點字面值。
- 字元常值，也稱為字串、字元字串或字元常數
- 日期時間 (datetime) 與間隔常值，搭配 datetime 資料類型使用。如需詳細資訊，請參閱 日期、時間和時間戳記常值 及 間隔資料類型和常值。

Null

如果列所在的欄遺漏、未知或不適用，則此欄為 null 值，或是包含 null。Null 可以出現在任何資料類型的欄位中，而且不受主索引鍵或 NOT NULL 限制的限制。Null 不等於 0 值，也不等於空字串。

任何包含 null 的數學表達式，所求得的結果一律為 null。給予 null 引數或運算元時，除了串連以外的所有運算子，都會傳回 null。

若要檢定 null，請使用比較條件 IS NULL 和 IS NOT NULL。由於 null 代表缺少資料，因此 null 不等於任何值，也不等於另一個 null。

資料類型

主題

- 多位元組字元

Amazon Redshift 所儲存或擷取的每個值，都具有資料類型，而資料類型具有一組固定的相關屬性。資料類型會在資料表建立時宣告，用來限制欄或引數可包含的一組值。

下表列出您可以在 Amazon Redshift 中使用的資料類型。

資料類型	Aliases	描述
SMALLINT	INT2	帶正負號的 2 位元組整數
INTEGER	INT、INT4	帶正負號的 4 位元組整數
BIGINT	INT8	帶正負號的 8 位元組整數
DECIMAL	NUMERIC	可選擇精確度 (有效位數) 的精確數值
REAL	FLOAT4	單精度浮點數
DOUBLE PRECISION	FLOAT8、FLOAT	雙精度浮點數
CHAR	CHARACTER、NCHAR、BPCHAR	固定長度的字元字串
VARCHAR	CHARACTER VARYING、NVARCHAR、TEXT	可變長度的字元字串 (使用者定義的限制)
DATE		日曆日期 (年、月、日)
TIME	TIME WITHOUT TIME ZONE	一天中的時間

資料類型	Aliases	描述
TIMETZ	TIME WITH TIME ZONE	一天中的時間，含時區
TIMESTAMP	TIMESTAMP WITHOUT TIME ZONE	日期和時間 (未使用時區)
TIMESTAMPTZ	TIMESTAMP WITH TIME ZONE	日期和時間 (包含時區)
INTERVAL YEAR TO MONTH		一年到月訂單的持續時間
INTERVAL DAY TO SECOND		從天到第二順序的持續時間
BOOLEAN	BOOL	邏輯布林值 (true/false)
HLLSKETCH		與 HyperLogLog 草圖一起使用的類型。
SUPER		超集資料類型，包含 Amazon Redshift 的所有純量類型，包括複雜類型 (例如 ARRAY 和 STRUCTS)。
VARBYTE	VARBINARY、BINARY VARYING	可變長度二進位值
GEOMETRY		空間資料
GEOGRAPHY		空間資料

> **Note**
>
> 如需 "char" (請注意，char 會用引號括住) 等不支援資料類型的相關資訊，請參閱不支援的 PostgreSQL 資料類型。

多位元組字元

VARCHAR 資料類型支援最多 4 個位元組的 UTF-8 多位元組字元，不支援 5 個位元組或更長的字元。若要針對包含多位元組字元的 VARCHAR 資料欄，計算其大小，請將字元數乘以每個字元的位元組數。例如，如果字串包含 4 個中文字，而每個字的長度是 3 個位元組，那麼您將需要使用 VARCHAR(12) 資料欄來儲存這個字串。

VARCHAR 資料類型不支援下列無效的 UTF-8 碼位：

0xD800 – 0xDFFF (位元組序列：ED A0 80 - ED BF BF)

CHAR 資料類型不支援多位元組字元。

數值類型

主題

- 整數類型
- DECIMAL 或 NUMERIC 類型
- 關於使用 128 位元 DECIMAL 或 NUMERIC 資料欄的備註
- 浮點類型
- 數值的計算
- 整數常值與浮點字面值
- 數值類型範例

數值資料類型包括整數、小數和符點數。

整數類型

使用 SMALLINT、INTEGER 和 BIGINT 資料類型來儲存各種範圍的整數。您不能儲存超出每種類型允許範圍的值。

名稱	儲存	範圍
SMALLINT 或 INT2	2 位元組	-32768 到 +32767
INTEGER、INT 或 INT4	4 位元組	-2147483648 到 +2147483647

名稱	儲存	範圍
BIGINT 或 INT8	8 位元組	-9223372036854775808 到 9223372036854775807

DECIMAL 或 NUMERIC 類型

使用 DECIMAL 或 NUMERIC 資料類型，以使用者定義的精確度來儲存數值。DECIMAL 和 NUMERIC 關鍵字可互換使用。在本文件中，小數是此資料類型的首選用詞。數值一詞通常是用來指稱整數、小數和浮點資料類型。

儲存	範圍
變數，未壓縮的 DECIMAL 類型最多 128 位元。	128 位元帶正負號的整數，具備最高 38 個位數的精確度。

藉由指定 precision 和 scale，來定義資料表中的 DECIMAL 欄：

```
decimal(precision, scale)
```

precision

整個值中有效位數的總數：小數點兩邊的位數數量。例如，數字 48.2891 的精確度 (有效位數) 為 6，小數位數為 4。如果未指定，預設的精確度為 18，最高精確度為 38。

如果在輸入值中，小數點左邊的位數數目，超過資料欄的精確度減去其小數位數，就無法將此值複製 (或插入或更新) 到資料欄中。此規則適用於超出資料欄定義範圍之外的任何值。例如，numeric(5,2) 欄的值，其允許的範圍為 -999.99 到 999.99。

scale

數值小數部分中，位於小數點右邊的小數位數數目。整數的小數位數為 0。在資料欄的規格中，小數位數的值必須小於或等於精確度的值。如果未指定，預設的小數位數為 0，最大的小數位數為 37。

如果載入資料表的輸入值，其小數位數大於資料欄的小數位數，則此值會四捨五入至指定的小數位數。例如，SALES 資料表中的 PRICEPAID 資料欄為 DECIMAL(8,2) 資料欄。如果將 DECIMAL(8,4) 值插入 PRICEPAID 資料欄，會將此值四捨五入為 2 個小數位數。

```
insert into sales
values (0, 8, 1, 1, 2000, 14, 5, 4323.8951, 11.00, null);

select pricepaid, salesid from sales where salesid=0;

pricepaid | salesid
----------+---------
4323.90 |       0
(1 row)
```

不過，從資料表所選取值的明確轉換結果，不會四捨五入。

> **Note**
>
> 可以插入 DECIMAL(19,0) 資料欄的正數值上限為 9223372036854775807 (2^{63} -1)。負數值上限為 -9223372036854775807。例如，如果試圖插入數值 9999999999999999999 (19 個 9)，將會造成溢位錯誤。無論小數點的位置何在，Amazon Redshift 可以表示為 DECIMAL 數值的最大字串是 9223372036854775807。例如，可以載入 DECIMAL(19,18) 資料欄的最大值為 9.223372036854775807。
>
> 這些規則是因為具有 19 個或更少有效位數有效位數的 DECIMAL 值會在內部儲存為 8 位元組整數，而具有 20 到 38 位有效位數的 DECIMAL 值則儲存為 16 位元組整數。

關於使用 128 位元 DECIMAL 或 NUMERIC 資料欄的備註

除非您確定應用程式需要該精確度，否則請勿任意指派最大有效位數給 DECIMAL 欄。128 位元值使用的磁碟空間是 64 位元值的兩倍，而且可能會減慢查詢執行時間。

浮點類型

使用 REAL 和 DOUBLE PRECISION 資料類型，以可變精確度來儲存數值。這些是不精確的類型，代表某些數值會以近似值儲存，因此在儲存和傳回特定值時，可能會造成些微的出入。如果您需要精確的儲存和計算 (例如貨幣金額)，請使用 DECIMAL 資料類型。

REAL 代表遵循二進位浮點數運算之 IEEE 標準 754 的單精確度浮點格式。它具有大約 6 位數的精確度，並且範圍約為 1E-37 到 1E+37。您也可以將此資料類型指定為 FLOAT4。

DOUBLE PRECISION 代表遵循二進位浮點數運算之 IEEE 標準 754 的雙精確度浮點格式。它具有大約 15 位數的精確度，並且範圍約為 1E-307 到 1E+308。您也可以將此資料類型指定為 FLOAT 或 FLOAT8。

除了普通的數值之外，浮點類型還有幾個特殊值。在 SQL 中使用這些值時，請使用單引號：

- NaN – not-a-number
- `Infinity`— 無窮大
- `-Infinity`— 負無窮大

例如，要在表`day_charge`的列 not-a-number 中插入，請`customer_activity`運行以下 SQL：

```
insert into customer_activity(day_charge) values('NaN');
```

數值的計算

在本文中，計算是指二進位數學運算：加、減、乘和除。本節說明這些運算預期的傳回類型，以及使用 DECIMAL 資料類型時，用來決定精確度與小數位數的特定公式。

在查詢處理作業期間計算數值時，可能會遇到無法進行計算的情況，而且查詢會傳回數值溢位錯誤。您也可能會遇到計算值的小數位數改變或出乎意料的情況。針對某些運算，您可以使用明確轉換 (類型提升) 或 Amazon Redshift 設定參數，來解決這些問題。

關於使用 SQL 函式進行類似計算的結果，詳細資訊請參閱 彙總函數。

計算的傳回類型

根據 Amazon Redshift 中所支援的一組數值資料類型，下表顯示了加法、減法、乘法和除法運算預期的傳回類型。表格左側的第一欄代表計算中的第一個運算元，最上面的列代表第二個運算元。

	INT2	INT4	INT8	DECIMAL	FLOAT4	FLOAT8
INT2	INT2	INT4	INT8	DECIMAL	FLOAT8	FLOAT8
INT4	INT4	INT4	INT8	DECIMAL	FLOAT8	FLOAT8
INT8	INT8	INT8	INT8	DECIMAL	FLOAT8	FLOAT8
DECIMAL	DECIMAL	DECIMAL	DECIMAL	DECIMAL	FLOAT8	FLOAT8

FLOAT4	FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT4	FLOAT8
FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT8	FLOAT8

計算出 DECIMAL 結果的精確度和小數位數

下表顯示摘要，說明在數學運算傳回 DECIMAL 結果時，用來計算結果精確度和小數位數的規則。在這個表格中，p1 和 s1 代表計算式中第一個運算元的精確度和小數位數，p2 和 s2 代表第二個運算元的精確度和小數位數。(無論這些計算如何，結果的最高精確度為 38、結果的最大小數位數為 38。)

作業	結果的精確度與小數位數
+ 或 -	擴展 = max(s1,s2) 精確度 = max(p1-s1,p2-s2)+1+scale
*	擴展 = s1+s2 精確度 = p1+p2+1
/	擴展 = max(4,s1+p2-s2+1) 精確度 = p1-s1+ s2+scale

例如，SALES 資料表中的 PRICEPAID 和 COMMISSION 資料欄都是 DECIMAL(8,2) 資料欄。如果將 PRICEPAID 除以 COMMISSION (或反過來)，會如下套用公式：

```
Precision = 8-2 + 2 + max(4,2+8-2+1)
= 6 + 2 + 9 = 17

Scale = max(4,2+8-2+1) = 9

Result = DECIMAL(17,9)
```

下列的計算是一般規則，適用於針對 DECIMAL 數值的運算 (使用 UNION、INTERSECT 和 EXCEPT 等集合運算子，或 COALESCE 和 DECODE 等函式)，計算出結果的精確度和小數位數：

```
Scale = max(s1,s2)
```

```
Precision = min(max(p1-s1,p2-s2)+scale,19)
```

例如，包含一個 DECIMAL(7,2) 資料欄的 DEC1 資料表，會與包含一個 DECIMAL(15,3) 資料欄的 DEC2 資料表聯結，以產生 DEC3 資料表。DEC3 的結構描述顯示，其資料欄會變成 NUMERIC(15,3) 資料欄。

```
create table dec3 as select * from dec1 union select * from dec2;
```

結果

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'dec3';

column |     type      | encoding | distkey | sortkey
-------+---------------+----------+---------+---------
c1     | numeric(15,3) | none     | f       | 0
```

在上述的範例中，會如下套用公式：

```
Precision = min(max(7-2,15-3) + max(2,3), 19)
= 12 + 3 = 15

Scale = max(2,3) = 3

Result = DECIMAL(15,3)
```

關於除法運算的備註

對於除法作業， divide-by-zero 條件會傳回錯誤。

在計算出精確度和小數位數之後，會套用 100 個小數位數的限制。如果計算結果的小數位數大於 100，除的結果會如下設定小數位數：

- 精確度 = precision - (scale - max_scale)
- 擴展 = max_scale

如果計算出的精確度大於最高精確度 (38)，則精確度會降低為 38，而小數位數會變成下列算式的結果：max((38 + scale - precision), min(4, 100))

溢位狀況

會針對所有數值運算檢查溢位。具有 19 個 (含) 以下有效位數 (精確度) 的 DECIMAL 資料，會儲存為 64 位元整數。具有 19 個 (含) 以上有效位數 (精確度) 的 DECIMAL 資料，會儲存為 128 位元整數。所有 DECIMAL 數值的最高精確度為 38、最大小數位數為 37。當數值超出這些限值時，會出現溢位錯誤，這會同時發生於中間的和最終的結果集：

- 當特定資料值不符合轉換函式所要求的精確度或指定的小數位數時，明確轉換作業會產生執行時期的溢位錯誤。例如，您無法從 SALES 資料表中的 PRICEPAID 資料欄 (DECIMAL(8,2) 資料欄) 轉換所有的數值，然後傳回 DECIMAL(7,3) 結果：

  ```
  select pricepaid::decimal(7,3) from sales;
  ERROR:  Numeric data overflow (result precision)
  ```

  由於 PRICEPAID 資料欄中的某些較大的數值無法轉換，因此發生了這項錯誤。
- 在乘法運算所產生的結果中，其小數位數是每個運算元小數位數的總和。例如，如果兩個運算元都有 4 個小數位數，結果的小數位數是 8 個，使得小數點的左邊只有 10 個位數。因此，在將兩個都擁有大量小數位數的大數值相乘時，就會相當容易產生溢位狀況。

  下列範例會導致溢位錯誤。

  ```
  SELECT CAST(1 AS DECIMAL(38, 20)) * CAST(10 AS DECIMAL(38, 20));
  ERROR: 128 bit numeric data overflow (multiplication)
  ```

  您可以使用除法而不是乘法來解決溢位錯誤。使用以下範例除以 1 除以原始除數。

  ```
  SELECT CAST(1 AS DECIMAL(38, 20)) / (1 / CAST(10 AS DECIMAL(38, 20)));
  +-----------+
  | ?column? |
  +-----------+
  | 10       |
  +-----------+
  ```

INTEGER 和 DECIMAL 類型的數值計算

當計算式中的其中一個運算元具有 INTEGER 資料類型，而另一個運算元為 DECIMAL，INTEGER 運算元會隱含轉換為 DECIMAL：

- INT2 (SMALLINT) 會轉換為 DECIMAL(5,0)

- INT4 (INTEGER) 會轉換為 DECIMAL(10,0)
- INT8 (BIGINT) 會轉換為 DECIMAL(19,0)

例如，如果將 DECIMAL(8,2) 資料欄 SALES.COMMISSION 乘以 SMALLINT 資料欄 SALES.QTYSOLD，此計算式會進行如下的轉換：

```
DECIMAL(8,2) * DECIMAL(5,0)
```

整數常值與浮點字面值

表示數字的常值或常數，可以是整數或浮點數。

整數常值

整數常值是數字 0 – 9 的序列，在數字前面可以選擇性地加上正號 (+) 或負號 (-)。

語法

```
[ + | - ] digit ...
```

範例

有效的整數包括下列範例：

```
23
-555
+17
```

浮點字面值

浮點字面值 (也稱為小數、數值或分數常值) 是數字的序列，可包含小數點，以及選擇性的包含指數標記。

語法

```
[ + | - ] digit ... [ . ] [ digit ...]
[ e | E [ + | - ] digit ... ]
```

引數

e | E

e 或 E 代表該數字是以科學記號指定。

範例

有效的浮點字面值包括下列範例：

```
3.14159
-37.
2.0e19
-2E-19
```

數值類型範例

CREATE TABLE 陳述式

下列的 CREATE TABLE 陳述式示範不同數值資料類型的宣告：

```
create table film (
film_id integer,
language_id smallint,
original_language_id smallint,
rental_duration smallint default 3,
rental_rate numeric(4,2) default 4.99,
length smallint,
replacement_cost real default 25.00);
```

試圖插入超出範圍的整數

下列的範例試圖將 33000 這個值插入 SMALLINT 資料欄。

```
insert into film(language_id) values(33000);
```

SMALLINT 的範圍是 -32768 到 +32767，因此 Amazon Redshift 會傳回錯誤。

```
An error occurred when executing the SQL command:
insert into film(language_id) values(33000)
```

```
ERROR: smallint out of range [SQL State=22003]
```

將小數值插入整數資料欄

下列的範例將小數值插入 INT 資料欄。

```
insert into film(language_id) values(1.5);
```

會插入此數值，但四捨五入為整數值 2。

成功地插入小數，因為其小數位數已經過四捨五入

下列的範例所插入的小數值，具有高於資料欄的精確度。

```
insert into film(rental_rate) values(35.512);
```

在此例中，將 35.51 這個值插入了資料欄。

試圖插入超出範圍的小數值

在此例中，350.10 這個值超出範圍。在 DECIMAL 資料欄中，數值的位數等於該資料欄的有效位數 (精確度) 減掉其小數位數 (RENTAL_RATE 資料欄為 4 減掉 2)。換句話說，DECIMAL(4,2) 資料欄允許的範圍是 -99.99 到 99.99。

```
insert into film(rental_rate) values (350.10);
ERROR:  numeric field overflow
DETAIL:  The absolute value is greater than or equal to 10^2 for field with precision
 4, scale 2.
```

將可變精確度的值插入 REAL 資料欄

下列的範例將可變精確度的值插入 REAL 資料欄。

```
insert into film(replacement_cost) values(1999999.99);

insert into film(replacement_cost) values(1999.99);

select replacement_cost from film;

+------------------+
| replacement_cost |
+------------------+
```

```
| 2000000          |
| 1999.99          |
+------------------+
```

1999999.99 值會轉換為 2000000，以符合 REAL 欄的精確度要求。數值 1999.99 會依原狀載入。

字元類型

主題

- 儲存與範圍
- CHAR 或 CHARACTER
- VARCHAR 或 CHARACTER VARYING
- NCHAR 與 NVARCHAR 類型
- TEXT 與 BPCHAR 類型
- 多餘空格的意義
- 字元類型範例

字元資料類型包括 CHAR (字元) 和 VARCHAR (可變長度字元)。

儲存與範圍

CHAR 和 VARCHAR 資料類型是以字元組而非字元來定義。CHAR 資料欄只能包含單位元組字元，因此 CHAR(10) 資料欄可包含最大長度為 10 位元組的字串。VARCHAR 可包含多位元組字元，每個字元最多 4 個位元組。例如，VARCHAR(12) 資料欄可包含 12 個單位元組的字元、6 個 2 位元組的字元、4 個 3 位元組的字元，或是 3 個 4 位元組的字元。

名稱	儲存	範圍 (資料欄的寬度)
CHAR、CHARACTER 或 NCHAR	字串的長度，包括多餘的空格 (如果有的話)	4096 位元組
VARCHAR、CHARACTER VARYING 或 NVARCHAR	4 位元組 + 字元的總位元組數，其中每個字元都可以是 1 到 4 個位元組。	65535 位元組 (64K -1)

名稱	儲存	範圍 (資料欄的寬度)
BPCHAR	轉換為固定長度 CHAR(256)。	256 位元組
TEXT	轉換為 VARCHAR(256)。	260 位元組

> **Note**
>
> CREATE TABLE 語法支援字元資料類型的 MAX 關鍵字。例如：
>
> ```
> create table test(col1 varchar(max));
> ```
>
> MAX 設定會分別將 CHAR 或 VARCHAR 的資料欄寬度，定義為 4096 或 65535 個位元組。

CHAR 或 CHARACTER

使用 CHAR 或 CHARACTER 資料欄來儲存固定長度的字串。這些字串會用空格填充，因此 CHAR(10) 資料欄一律會佔 10 個位元組的儲存空間。

```
char(10)
```

未指定長度規格的 CHAR 資料欄，會變成 CHAR(1) 資料欄。

VARCHAR 或 CHARACTER VARYING

使用 VARCHAR 或 CHARACTER VARYING 資料欄，來儲存具有固定限制的可變長度字串。這些字串並未使用空格填充，因此，VARCHAR(120) 資料欄最多可包含 120 個單位元組的字元、60 個 2 位元組的字元、40 個 3 位元組的字元，或是 30 個 4 位元組的字元。

```
varchar(120)
```

如果您在建立資料表陳述式中使用沒有長度說明符的 VARCHAR 資料類型，預設長度為 256。如果在運算式中使用，系統會使用輸入運算式來確定輸出的大小 (最多 65535)。

NCHAR 與 NVARCHAR 類型

您可以建立具有 NCHAR 和 NVARCHAR 類型的資料欄 (也稱為 NATIONAL CHARACTER 和 NATIONAL CHARACTER VARYING 類型)。這些類型會分別轉換為 CHAR 和 VARCHAR 類型，然後以指定的位元組數目儲存。

未指定長度規格的 NCHAR 資料欄，會轉換為 CHAR(1) 資料欄。

未指定長度規格的 NVARCHAR 資料欄，會轉換為 VARCHAR(256) 資料欄。

TEXT 與 BPCHAR 類型

您可以建立包含 TEXT 欄的 Amazon Redshift 資料表，但是此欄會轉換為 VARCHAR(256) 欄，接受最多 256 個字元的可變長度值。

您可以建立具備 BPCHAR (空格填充字元) 類型的 Amazon Redshift 欄，Amazon Redshift 會轉換為固定長度的 CHAR(256) 欄。

多餘空格的意義

CHAR 和 VARCHAR 資料類型都會儲存長度最多 n 個位元組的字串。如果試圖將較長的字串，儲存到具有這些類型的資料欄中，將會造成錯誤 (除非多出來的字串全部都是空格 (空白)，此時字串會被截斷至最大長度)。如果字串短於最大長度，CHAR 值會以空格填充，但 VARCHAR 值則會儲存不含空格的字串。

CHAR 值中的多餘空格在語義上一律不具有意義。這些空格會在您比較兩個 CHAR 值時被忽略、不列入 LENGTH 的計算中，而且會在您將 CHAR 值轉換為另一種字串類型時移除。

在比較值時，VARCHAR 和 CHAR 值中的多餘空格，在語義上會視為不具意義。

長度的計算會傳回 VARCHAR 字元字串的長度，其中也包含多餘的空格。多餘的空格不會列入固定長度字元字串的長度計算。

字元類型範例

CREATE TABLE 陳述式

下列的 CREATE TABLE 陳述式示範 VARCHAR 和 CHAR 資料類型的使用：

```
create table address(
address_id integer,
```

```
address1 varchar(100),
address2 varchar(50),
district varchar(20),
city_name char(20),
state char(2),
postal_code char(5)
);
```

下列的範例使用此資料表。

可變長度字元字串中的多餘空格

由於 ADDRESS1 是 VARCHAR 資料欄，因此在第二個插入的地址中，多餘的空格在語義上是無關緊要的。換句話說，下列這兩個插入的地址是相符合的。

```
insert into address(address1) values('9516 Magnolia Boulevard');

insert into address(address1) values('9516 Magnolia Boulevard  ');
```

```
select count(*) from address
where address1='9516 Magnolia Boulevard';

count
-------
2
(1 row)
```

如果 ADDRESS1 資料欄原本是 CHAR 資料欄，而且插入了相同的值，則 COUNT(*) 查詢會將字元字串視為相同，並傳回 2。

LENGTH 函數的結果

LENGTH 函式會辨識 VARCHAR 資料欄中的多餘空格：

```
select length(address1) from address;

length
--------
23
25
(2 rows)
```

在 CITY_NAME 資料欄 (CHAR 資料欄) 中的 Augusta 值，無論輸入字串中是否有任何多餘的空格，一律會傳回 7 個字元的長度。

超過資料欄長度的值

字元字串不會為了配合資料欄宣告的寬度而遭到截斷：

```
insert into address(city_name) values('City of South San Francisco');
ERROR: value too long for type character(20)
```

這個問題的解決方法，是將值轉換為符合資料欄的大小：

```
insert into address(city_name)
values('City of South San Francisco'::char(20));
```

在這個例子中，字串 (City of South San Fr) 的前 20 個字元會載入資料欄。

日期時間 (Datetime) 類型

主題

- 儲存與範圍
- DATE
- TIME
- TIMETZ
- TIMESTAMP
- TIMESTAMPTZ
- 日期時間 (Datetime) 類型範例
- 日期、時間和時間戳記常值
- 間隔資料類型和常值

日期時間 (Datetime) 資料類型包含 DATE、TIME、TIMETZ、TIMESTAMP 與 TIMESTAMPTZ。

儲存與範圍

名稱	儲存	範圍	解析度
DATE	4 位元組	4713 BC 到 294276 AD	1 天

名稱	儲存	範圍	解析度
TIME	8 位元組	00:00:00 至 24:00:00	1 毫秒
TIMETZ	8 位元組	00:00:00+1459 至 00:00:00+1459	1 毫秒
TIMESTAMP	8 位元組	4713 BC 到 294276 AD	1 毫秒
TIMESTAMPTZ	8 位元組	4713 BC 到 294276 AD	1 毫秒

DATE

使用 DATE 資料類型來儲存不含時間戳記的簡單日曆日期。

TIME

TIME 是 TIME WITHOUT TIME ZONE 的別名。

使用 TIME 資料類型來儲存一天中的時間。

TIME 欄可針對小數秒數，儲存精確度最高 6 位數的數值。

在使用者資料表和 Amazon Redshift 系統資料表中，TIME 值預設皆採用世界標準時間 (UTC)。

TIMETZ

TIMETZ 是 TIME WITH TIME ZONE 的別名。

使用 TIMETZ 資料類型來儲存具有時區的一天中的時間。

TIMETZ 欄可針對小數秒數，儲存精確度最高 6 位數的數值。

根據預設，TIMTZ 值在使用者資料表和 Amazon Redshift 系統資料表中皆採用 UTC。

TIMESTAMP

TIMESTAMP 是 TIMESTAMP WITHOUT TIME ZONE 的別名。

使用 TIMESTAMP 資料類型來儲存完整的時間戳記值，其中包含日期和當日的時間。

TIMESTAMP 欄可針對小數秒數，儲存精確度最高 6 位數的數值。

如果將日期插入 TIMESTAMP 欄，或具有部分時間戳記值的日期，則該值會隱含轉換為完整時間戳記值。此完整時間戳記值對於缺少的小時、分鐘和秒具有預設值 (00)。輸入字串中的時區值會遭到忽略。

根據預設，TIMESTAMP 值在使用者資料表和 Amazon Redshift 系統資料表中皆採用 UTC。

TIMESTAMPTZ

TIMESTAMPTZ 是 TIMESTAMP WITH TIME ZONE 的別名。

使用 TIMESTAMPTZ 資料類型來輸入完整的時間戳記值，其中包含日期、當日的時間和時區。當輸入值包含時區時，Amazon Redshift 會使用時區來將該值轉換為 UTC，並儲存 UTC 值。

若要查看受支援時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```

若要查看受支援時區縮寫的清單，請執行下列命令。

```
select pg_timezone_abbrevs();
```

在 IANA 時區資料庫中，也提供了關於時區的最新資訊。

下表提供時區格式的範例。

格式	範例
dd mon hh:mi:ss yyyy tz	17 Dec 07:37:16 1997 PST
mm/dd/yyyy hh:mi:ss.ss tz	12/17/1997 07:37:16.00 PST
mm/dd/yyyy hh:mi:ss.ss tz	12/17/1997 07:37:16.00 美國時間/太平洋時區
yyyy-mm-dd H: 三分鐘:SS+/-TZ	1997-12-17 07:37:16-08
dd.mm.yyyy hh:mi:ss tz	17.12.1997 07:37:16.00 PST

TIMESTAMPTZ 欄可針對小數秒數，儲存精確度最高 6 位數的數值。

如果將日期插入 TIMESTAMPTZ 欄，或具有部分時間戳記的日期，則該值會隱含轉換為完整時間戳記值。此完整時間戳記值對於缺少的小時、分鐘和秒具有預設值 (00)。

TIMESTAMPTZ 值在使用者資料表中採用 UTC。

日期時間 (Datetime) 類型範例

接下來，您可以找到使用 Amazon Redshift 支援的日期時間類型的範例。

日期範例

以下範例插入具有不同格式的日期並顯示輸出。

```
create table datetable (start_date date, end_date date);
```

```
insert into datetable values ('2008-06-01','2008-12-31');

insert into datetable values ('Jun 1,2008','20081231');
```

```
select * from datetable order by 1;

start_date |  end_date
-----------------------
2008-06-01 | 2008-12-31
2008-06-01 | 2008-12-31
```

如果將時間戳記值插入 DATE 資料欄，會略過時間的部分，只載入日期。

時間範例

以下範例插入具有不同格式的 TIME 和 TIMETZ 值並顯示輸出。

```
create table timetable (start_time time, end_time timetz);
```

```
insert into timetable values ('19:11:19','20:41:19 UTC');
insert into timetable values ('191119', '204119 UTC');
```

```
select * from timetable order by 1;
start_time |  end_time
-----------------------
```

```
 19:11:19  | 20:41:19+00
 19:11:19  | 20:41:19+00
```

時間戳記範例

如果將日期插入 TIMESTAMP 或 TIMESTAMPTZ 資料欄，則時間會預設為午夜。例如，如果插入常值 20081231，則儲存的值為 2008-12-31 00:00:00。

若要變更目前工作階段的時區，請利用 SET 指令來設定 timezone 組態參數。

下列範例會插入具有不同格式的時間戳記，並顯示產生的資料表。

```
create table tstamp(timeofday timestamp, timeofdaytz timestamptz);

insert into tstamp values('Jun 1,2008  09:59:59', 'Jun 1,2008 09:59:59 EST' );
insert into tstamp values('Dec 31,2008 18:20','Dec 31,2008 18:20');
insert into tstamp values('Jun 1,2008  09:59:59 EST', 'Jun 1,2008 09:59:59');

SELECT * FROM tstamp;

+---------------------+-----------------------+
|      timeofday      |      timeofdaytz      |
+---------------------+-----------------------+
| 2008-06-01 09:59:59 | 2008-06-01 14:59:59+00 |
| 2008-12-31 18:20:00 | 2008-12-31 18:20:00+00 |
| 2008-06-01 09:59:59 | 2008-06-01 09:59:59+00 |
+---------------------+-----------------------+
```

日期、時間和時間戳記常值

以下是 Amazon Redshift 所支援日期、時間和時間戳記常值的規則。

日期

下列輸入日期是您可以載入 Amazon Redshift 表格之 DATE 資料類型之常值的所有有效範例。假設預設 MDY DateStyle 模式有效。此模式表示在字串中月份值位於日期值之前，例如 1999-01-08 和 01/02/00。

> **Note**
>
> 載入資料表時，日期或時間戳記常值必須用引號括住。

輸入的日期	完整日期
January 8, 1999	January 8, 1999
1999-01-08	January 8, 1999
1/8/1999	January 8, 1999
01/02/00	2000 年 1 月 2 日
2000-Jan-31	2000 年 1 月 31 日
Jan-31-2000	2000 年 1 月 31 日
31-Jan-2000	2000 年 1 月 31 日
20080215	2008 年 2 月 15 日
080215	2008 年 2 月 15 日
2008.366	2008 年 12 月 31 日 (日期的 3 位數部分必須介於 001 到 366 之間)

Times

下列輸入時間是您可以載入 Amazon Redshift 資料表的時間和 TIMETZ 資料類型之常值的所有有效範例。

輸入時間	說明 (時間的部分)
04:05:06.789	4:05 AM 又 6.789 秒
04:05:06	4:05 AM 又 6 秒
04:05	4:05 AM 整
040506	4:05 AM 又 6 秒
04:05 AM	4:05 AM 整；AM 為選用

輸入時間	說明 (時間的部分)
04:05 PM	4:05 PM 整；小時值必須小於 12。
16:05	4:05 PM 整

時間戳記

下列輸入時間戳記是您可以載入 Amazon Redshift 資料表的時間戳記和時間戳記資料類型之文字時間值的所有有效範例。有效的日期常值全都可以和下列的時間常值合併。

輸入的時間戳記 (串接的日期和時間)	說明 (時間的部分)
20080215 04:05:06.789	4:05 AM 又 6.789 秒
20080215 04:05:06	4:05 AM 又 6 秒
20080215 04:05	4:05 AM 整
20080215 040506	4:05 AM 又 6 秒
20080215 04:05 AM	4:05 AM 整；AM 為選用
20080215 04:05 PM	4:05 PM 整；小時值必須小於 12。
20080215 16:05	4:05 PM 整
20080215	午夜 (預設)

特殊的日期時間 (Datetime) 值

下列的特殊值可做為日期時間 (datetime) 常值和日期函式的引數使用。這些值需使用單引號，而且會在查詢處理作業進行期間，轉換為一般的時間戳記值。

特殊值	描述
now	轉換為目前交易的開始時間，並傳回毫秒精確度的時間戳記。

特殊值	描述
today	轉換為適當的日期，並傳回時間戳記，其中時間的部分全部以 0 表示。
tomorrow	轉換為適當的日期，並傳回時間戳記，其中時間的部分全部以 0 表示。
yesterday	轉換為適當的日期，並傳回時間戳記，其中時間的部分全部以 0 表示。

下列範例顯示 now 和 today 如何與 DATEADD 函數搭配使用。

```
select dateadd(day,1,'today');

date_add
---------------------
2009-11-17 00:00:00
(1 row)

select dateadd(day,1,'now');

date_add
----------------------------
2009-11-17 10:45:32.021394
(1 row)
```

間隔資料類型和常值

您可以使用間隔資料類型，以單位儲存時間的持續時間seconds，例如minuteshours、days、months、和years。間隔資料類型和常值可用於日期時間計算，例如，將間隔新增至日期和時間戳記、加總間隔，以及從日期或時間戳記減去間隔。間隔常值可做為資料表中間隔資料類型資料行的輸入值。

間隔數據類型的語法

若要指定間隔資料類型，以儲存以年和月為單位的持續時間：

```
INTERVAL year_to_month_qualifier
```

若要指定間隔資料類型，以儲存以天、小時、分鐘和秒為單位的持續時間：

```
INTERVAL day_to_second_qualifier [ (fractional_precision) ]
```

間隔文字的語法

若要指定間隔常值來定義以年和月為單位的持續時間：

```
INTERVAL quoted-string year_to_month_qualifier
```

若要指定間隔常值，以定義以天、小時、分鐘和秒為單位的持續時間：

```
INTERVAL quoted-string day_to_second_qualifier [ (fractional_precision) ]
```

引數

引用字符串

指定正數或負數值，將數量和日期時間單位指定為輸入字串。如果引號字串只包含一個數字，那麼 Amazon Redshift 會從年份 _ 月份限定詞或 day_to_second_限定符中確定單位。例如，'23' MONTH表示1 year 11 months、'-2' DAY代表-2 days 0 hours 0 minutes 0.0 seconds1 year 2 months、'1-2' MONTH代表和'13 day 1 hour 1 minute 1.123 seconds' SECOND表示13 days 1 hour 1 minute 1.123 seconds。如需間隔輸出格式的詳細資訊，請參閱間隔樣式。

年至月限定元

指定間隔的範圍。如果您使用限定詞並建立時間單位小於限定詞的間隔，Amazon Redshift 會截斷並捨棄間隔的較小部分。年至月限定詞的有效值為：

- YEAR
- MONTH
- YEAR TO MONTH

日至第二個限定詞

指定間隔的範圍。如果您使用限定詞並建立時間單位小於限定詞的間隔，Amazon Redshift 會截斷並捨棄間隔的較小部分。日至秒限定詞的有效值如下：

- DAY
- HOUR
- MINUTE

- SECOND
- DAY TO HOUR
- DAY TO MINUTE
- DAY TO SECOND
- HOUR TO MINUTE
- HOUR TO SECOND
- MINUTE TO SECOND

INTERVAL 常值的輸出會截斷為指定的最小間隔元件。例如，使用分鐘限定詞時，Amazon Redshift 會捨棄小於分鐘的時間單位。

```
select INTERVAL '1 day 1 hour 1 minute 1.123 seconds' MINUTE
```

產生的值會被截斷為'1 day 01:01:00'。

分數精度

可選參數，指定間隔中允許的小數位數。只有當你的間隔包含第二個時，才應該指定分數精度參數。例如，SECOND(3)建立只允許三個小數位數的間隔，例如 1.234 秒。小數位數的最大數目為 6。

會話配置interval_forbid_composite_literals確定當兩個年到月和日到第二部分指定的間隔時是否返回錯誤。如需詳細資訊，請參閱 間隔 _ 禁止複合 _ 文字。

间隔算术

您可以將間隔值與其他日期時間值搭配使用來執行算術運算。下表說明可用的作業，以及每項作業所產生的資料類型。例如，當你添加一個interval到一個結果是一date個date如果它是一個年到月的間隔，如果它是一個時間戳，如果它是一個日到第二個間隔。

		日期	時間戳記	Interval (間隔)	數值
Interval (間隔)	-	N/A	N/A	Interval (間隔)	N/A
	+	日期	日期/時間戳	Interval (間隔)	N/A

		日期	時間戳記	Interval (間隔)	數值
	*	N/A	N/A	N/A	Interval (間隔)
	/	N/A	N/A	N/A	Interval (間隔)
日期	-	數值	Interval (間隔)	日期/時間戳	日期
	+	N/A	N/A	N/A	N/A
Timestamp	-	Interval (間隔)	Interval (間隔)	時間戳記	時間戳記
	+	N/A	N/A	N/A	N/A

間隔樣式

您可以使用 SQL the section called “SET” 指令來變更間隔值的輸出顯示格式。當您在 SQL 中使用間隔資料類型時，請將其轉換為文字以查看預期的間隔樣式，例如，`YEAR TO MONTH::text`。設定值的可用`IntervalStyle`值為：

- `postgres`-遵循 PostgreSQL 風格。此為預設值。
- `postgres_verbose`— 遵 PostgreSQL 的樣式。
- `sql_standard`— 遵循 SQL 標準間隔常值樣式。

以下命令將間隔樣式設置為`sql_standard`。

```
SET IntervalStyle to 'sql_standard';
```

後輸出格式

以下是`postgres`間隔樣式的輸出格式。每個數值可以是負數。

```
'<numeric> <unit> [, <numeric> <unit> ...]'
```

```
select INTERVAL '1-2' YEAR TO MONTH::text

varchar
---------------
1 year 2 mons
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text

varchar
------------------
1 day 02:03:04.5678
```

詳細資訊輸出格式

詳細語法類似於 Postgres，但 postgres_verbose 輸出也包含時間單位。

```
'[@] <numeric> <unit> [, <numeric> <unit> ...] [direction]'
```

```
select INTERVAL '1-2' YEAR TO MONTH::text

varchar
-----------------
@ 1 year 2 mons
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text

varchar
---------------------------
@ 1 day 2 hours 3 mins 4.56 secs
```

sql_ 標準輸出格式

年到月的間隔值的格式如下。在間隔之前指定負號表示間隔為負值，並套用至整個間隔。

```
'[-]yy-mm'
```

間隔日到第二個值的格式如下。

```
'[-]dd hh:mm:ss.ffffff'
```

```
SELECT INTERVAL '1-2' YEAR TO MONTH::text

varchar
-------
1-2
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text

varchar
---------------
1 2:03:04.5678
```

區間資料類型的範例

下列範例示範如何搭配資料表使用 INTERVAL 資料類型。

```
create table sample_intervals (y2m interval month, h2m interval hour to minute);
insert into sample_intervals values (interval '20' month, interval '2 days
 1:1:1.123456' day to second);
select y2m::text, h2m::text from sample_intervals;


      y2m      |       h2m
---------------+-----------------
 1 year 8 mons | 2 days 01:01:00
```

```
update sample_intervals set y2m = interval '2' year where y2m = interval '1-8' year to
 month;
select * from sample_intervals;

   y2m   |       h2m
---------+-----------------
 2 years | 2 days 01:01:00
```

```
delete from sample_intervals where h2m = interval '2 1:1:0' day to second;
select * from sample_intervals;

 y2m | h2m
-----+-----
```

間隔常值的範例

下列範例會在將間隔型式設定為的情況下執行postgres。

下面的例子演示了如何創建一個 1 年的間隔文字。

```
select INTERVAL '1' YEAR

intervaly2m
---------------
1 years 0 mons
```

如果您指定的引號字串超過限定詞，剩餘的時間單位會從間隔中截斷。在下列範例中，13 個月的間隔會變成 1 年又 1 個月，但由於 YEAR 限定詞，剩餘的 1 個月則會被排除。

```
select INTERVAL '13 months' YEAR

intervaly2m
---------------
1 years 0 mons
```

如果您使用的限定詞小於間隔字串，則會包含剩餘的單位。

```
select INTERVAL '13 months' MONTH

intervaly2m
---------------
1 years 1 mons
```

指定間隔中的精確度會將小數位數截斷為指定的精確度。

```
select INTERVAL '1.234567' SECOND (3)

intervald2s
--------------------------------
0 days 0 hours 0 mins 1.235 secs
```

如果您沒有指定精確度，Amazon Redshift 會使用 6 的最大精確度。

```
select INTERVAL '1.23456789' SECOND
```

```
intervald2s
---------------------------------
0 days 0 hours 0 mins 1.234567 secs
```

下面的實例演示了如何創建一個範圍間隔。

```
select INTERVAL '2:2' MINUTE TO SECOND

intervald2s
-----------------------------
0 days 0 hours 2 mins 2.0 secs
```

限定詞會指定您指定的單位。例如，即使下列範例使用與前一個範例相同的引用字串「2:2」，Amazon Redshift 仍會辨識出由於限定詞而使用不同的時間單位。

```
select INTERVAL '2:2' HOUR TO MINUTE

intervald2s
-----------------------------
0 days 2 hours 2 mins 0.0 secs
```

還支持每個單元的縮寫和複數。例如，5s5 second、和5 seconds是等效的間隔。支援的單位包括年、月、小時、分鐘和秒。

```
select INTERVAL '5s' SECOND

intervald2s
-----------------------------
0 days 0 hours 0 mins 5.0 secs
```

```
select INTERVAL '5 HOURS' HOUR

intervald2s
-----------------------------
0 days 5 hours 0 mins 0.0 secs
```

```
select INTERVAL '5 h' HOUR

intervald2s
-----------------------------
```

```
0 days 5 hours 0 mins 0.0 secs
```

不含限定詞語法的間隔常值範例

> **Note**
>
> 下列範例示範如何使用不含YEAR TO MONTH或DAY TO SECOND限定詞的間隔常值。如需將建議的間隔常值與限定詞搭配使用的詳細資訊，請參閱間隔資料類型和常值。

使用間隔常值來表示指定的時間期間，例如 12 hours 或 6 months。您可以在需要表示日期時間的條件和表達式中，使用這些間隔常值。

間隔常值是以 INTERVAL 關鍵字與數值數量和支援日期部分的組合來表示，例如INTERVAL '7 days'或INTERVAL '59 minutes'。您可以串連幾個數量和單位，來組成更精確的間隔時間，例如：INTERVAL '7 days, 3 hours, 59 minutes'。也支援每種單位的縮寫和複數，例如 5 s、5 second 和 5 seconds 是相同的間隔時間。

如果未指定日期部分，則間隔值代表秒。您可以指定小數格式的數量值 (例如：0.5 days)。

下列範例顯示不同間隔值的一連串計算。

以下內容為指定日期增加 1 秒。

```
select caldate + interval '1 second' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 00:00:01
(1 row)
```

以下內容為指定日期增加 1 分鐘。

```
select caldate + interval '1 minute' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 00:01:00
(1 row)
```

以下內容為指定日期增加 3 個小時又 35 分鐘。

```
select caldate + interval '3 hours, 35 minutes' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 03:35:00
(1 row)
```

以下內容為指定日期增加 52 週。

```
select caldate + interval '52 weeks' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2009-12-30 00:00:00
(1 row)
```

以下內容為指定日期增加 1 週 1 小時 1 分鐘又 1 秒。

```
select caldate + interval '1w, 1h, 1m, 1s' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2009-01-07 01:01:01
(1 row)
```

以下內容為指定指定日期增加 12 個小時 (半天)。

```
select caldate + interval '0.5 days' as dateplus from date
where caldate='12-31-2008';
dateplus
---------------------
2008-12-31 12:00:00
(1 row)
```

以下內容從 2023 年 2 月 15 日減去 4 個月，結果是 2022 年 10 月 15 日。

```
select date '2023-02-15' - interval '4 months';

?column?
---------------------
2022-10-15 00:00:00
```

以下內容從 2023 年 3 月 31 日減去 4 個月，結果是 2022 年 11 月 30 日。計算有將一個月中的天數納入考量。

```
select date '2023-03-31' - interval '4 months';

?column?
---------------------
2022-11-30 00:00:00
```

布林值 (Boolean) 類型

使用 BOOLEAN 資料類型，在單位元組資料欄中儲存 true 和 false 值。下表說明 Boolean 值的三種可能狀態，以及導致狀態的字面值。無論輸入的字串為何，Boolean 資料欄都會分別將「t」和「f」儲存和輸出為 true 與 false。

State	有效的常值	儲存
True	`TRUE 't' 'true' 'y' 'yes' '1'`	1 位元組
False	`FALSE 'f' 'false' 'n' 'no' '0'`	1 位元組
不明	`NULL`	1 位元組

您只可以透過 WHERE 子句中述詞的形式使用 IS 比較來檢查布林值。您無法使用 IS 比較來搭配 SELECT 清單中的布林值。

範例

您可以使用 BOOLEAN 資料欄，針對 CUSTOMER 資料表中的每個客戶，儲存其「作用中/非作用中」狀態。

```
create table customer(
custid int,
active_flag boolean default true);
```

```
insert into customer values(100, default);
```

```
select * from customer;
custid | active_flag
-------+-------------
   100 | t
```

如果在 CREATE TABLE 陳述式中未指定預設值 (true 或 false)，則插入預設值代表插入 null。

在此範例中，查詢會從 USERS 資料表中，選取喜歡運動但不喜歡戲劇的使用者：

```
select firstname, lastname, likesports, liketheatre
from users
where likesports is true and liketheatre is false
order by userid limit 10;

firstname |  lastname  | likesports | liketheatre
----------+------------+------------+-------------
Lars      | Ratliff    | t          | f
Mufutau   | Watkins    | t          | f
Scarlett  | Mayer      | t          | f
Shafira   | Glenn      | t          | f
Winifred  | Cherry     | t          | f
Chase     | Lamb       | t          | f
Liberty   | Ellison    | t          | f
Aladdin   | Haney      | t          | f
Tashya    | Michael    | t          | f
Lucian    | Montgomery | t          | f
(10 rows)
```

下列範例中的查詢會從 USERS 資料表中選取不確定是否喜歡搖滾樂的使用者。

```
select firstname, lastname, likerock
from users
where likerock is unknown
order by userid limit 10;

firstname | lastname | likerock
----------+----------+----------
Rafael    | Taylor   |
Vladimir  | Humphrey |
```

```
Barry     | Roy      |
Tamekah   | Juarez   |
Mufutau   | Watkins  |
Naida     | Calderon |
Anika     | Huff     |
Bruce     | Beck     |
Mallory   | Farrell  |
Scarlett  | Mayer    |
(10 rows)
```

下列範例會傳回錯誤，因為它在 SELECT 清單中使用 IS 比較。

```
select firstname, lastname, likerock is true as "check"
from users
order by userid limit 10;

[Amazon](500310) Invalid operation: Not implemented
```

下列範例會成功，因為它在 SELECT 清單中使用等於比較 (=) 而非 IS 比較。

```
select firstname, lastname, likerock = true as "check"
from users
order by userid limit 10;

firstname | lastname  | check
----------+-----------+------
Rafael    | Taylor    |
Vladimir  | Humphrey  |
Lars      | Ratliff   | true
Barry     | Roy       |
Reagan    | Hodge     | true
Victor    | Hernandez | true
Tamekah   | Juarez    |
Colton    | Roy       | false
Mufutau   | Watkins   |
Naida     | Calderon  |
```

HLLSKETCH 類型

對 HyperLogLog 草圖使用 HLLSKETCH 資料類型。Amazon Redshift 支援稀疏或密集的 HyperLogLog 草圖表示。草圖一開始是稀疏的，當密集格式更有效地減少使用的記憶體佔用時，會切換到密集。

當以下列 JSON 格式匯入、匯出或列印草圖時，Amazon Redshift 會自動轉換稀疏 HyperLogLog 草圖。

```
{"logm":15,"sparse":{"indices":[4878,9559,14523],"values":[1,2,1]}}
```

Amazon Redshift 使用 Base64 格式的字串表示來表示密集 HyperLogLog 的草圖。

Amazon Redshift 使用以下 Base64 格式的字串表示來表示密集 HyperLogLog 的草圖。

```
"ABAABA..."
```

在 raw 壓縮中使用時，HLLSKETCH 物件的最大大小為 24,580 個位元組。

SUPER 類型

使用 SUPER 資料類型將半結構化資料或文件儲存為值。

半結構化資料不符合 SQL 資料庫中使用的關聯式資料模型的剛性和表格結構。它包含參考資料中不同實體的標籤。它們可以包含複雜值，例如陣列、巢狀結構，以及與序列化格式相關聯的其他複雜結構，例如 JSON。SUPER 資料類型是一組無結構描述陣列和結構值，包含 Amazon Redshift 的所有其他純量類型。

超級數據類型最多支持 16 MB 的數據的單個 SUPER 對象。如需 SUPER 資料類型的相關資訊，包括在資料表中的實作範例，請參閱在 Amazon Redshift 中擷取和查詢半結構化資料。

大於 1MB 的 SUPER 物件只能從下列檔案格式中擷取：

- Parquet
- JSON
- TEXT
- CSV

SUPER 資料類型具有下列屬性：

- Amazon Redshift 純量值：
  - Null
  - 布林值
  - 數字，例如 smallint、integer、bigint、decimal 或浮點數 (例如 float4 或 float8)
  - 字串值，如 varchar 或 char

- 複雜值：
  - 值的陣列，包括純量或複雜
  - 一種結構，也稱為元組或物件，它是屬性名稱和值 (純量或複雜) 的映射

這兩種類型的複雜值中的任何一種都包含它們自己的純量或複雜值，而沒有任何正規性限制。

SUPER 資料類型支援在無結構描述形式的半結構化資料的持久性。儘管階層式資料模型可能會發生變化，但舊版本的資料可以共存於同一 SUPER 欄中。

Amazon Redshift 使用 PartiQL 啟用陣列和結構的導覽功能。Amazon Redshift 也會使用 PartiQL 語法來迭代 SUPER 陣列。如需詳細資訊，請參閱 Navigation (導覽) 及 解除巢狀化查詢。

Amazon Redshift 會使用動態類型來處理無結構描述的 SUPER 資料，無需在查詢中使用資料類型之前先宣告資料類型。如需詳細資訊，請參閱 動態類型。

您可以將動態資料遮罩政策套用至 SUPER 類型資料欄路徑上的 `scalar` 值。如需動態資料遮罩的詳細資訊，請參閱 動態資料遮罩。如需將動態資料遮罩與 SUPER 資料類型搭配使用的詳細資訊，請參閱 搭配 SUPER 資料類型路徑使用動態資料遮罩。

VARBYTE 類型

使用 VARBYTE、VARBINARY 或 BINARY VARYING 欄來儲存具有固定限制的可變長度二進位值。

```
varbyte [ (n) ]
```

最大字節數 (n) 的範圍可以從 1-16,777,216。預設值為 64,000。

可以使用 VARBYTE 資料類型的一些範例如下：

- 在 VARBYTE 欄上聯結資料表。
- 建立包含 VARBYTE 欄的具體化視觀表。支援包含 VARBYTE 欄的具體化視觀表累加式重新整理。但是，VARBYTE 欄上的 COUNT、MIN 和 MAX 和 GROUP BY 以外的彙總函數不支援累加式重新整理。

為了確保所有位元組都是可列印字元，Amazon Redshift 使用十六進位格式來列印 VARBYTE 值。例如，下列 SQL 會將十六進位字串 6162 轉換成二進位值。即使傳回的值是二進位值，結果仍會列印為十六進位 6162。

```
select from_hex('6162');
```

```
 from_hex
----------
 6162
```

Amazon Redshift 支援在 VARBYTE 和以下資料類型之間進行轉換：

- CHAR
- VARCHAR
- SMALLINT
- INTEGER
- BIGINT

使用 CHAR 和 VARCHAR 進行轉換時，將使用 UTF-8 格式。如需有關 UTF-8 格式的相關資訊，請參閱 TO_VARBYTE。從 SMALLINT、INTEGER 和 BIGINT 進行轉換時，會保留原始資料類型的位元組數目。這對於 SMALLINT 來說是兩個位元組、對於 INTEGER 來說是四個位元組、對於 BIGINT 來說是八個位元組。

下列 SQL 陳述式將 VARCHAR 字串轉換為 VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 616263。

```
select 'abc'::varbyte;

 varbyte
---------
 616263
```

下列 SQL 陳述式將欄中的 CHAR 值轉換為 VARBYTE。這個範例會建立一個包含 CHAR(10) 欄 (c) 的資料表，插入長度小於 10 的字元值。產生的轉換將結果以空格字元 (hex'20') 填入定義的欄大小。即使傳回的值是二進位值，結果仍會列印為十六進位。

```
create table t (c char(10));
insert into t values ('aa'), ('abc');
select c::varbyte from t;
          c
----------------------
 6161202020202020202020
 6162632020202020202020
```

下列 SQL 陳述式將 SMALLINT 字串轉換為 VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 0005，即兩個位元組或四個十六進位字元。

```
select 5::smallint::varbyte;

 varbyte
---------
 0005
```

下列 SQL 陳述式將 INTEGER 轉換為 VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 00000005，即四個位元組或八個十六進位字元。

```
select 5::int::varbyte;

 varbyte
----------
 00000005
```

下列 SQL 陳述式將 BIGINT 轉換為 VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 0000000000000005，即八個位元組或 16 個十六進位字元。

```
select 5::bigint::varbyte;

     varbyte
------------------
 0000000000000005
```

支援 VARBYTE 資料類型的 Amazon Redshift 功能包括：

- VARBYTE 運算子
- CONCAT
- LEN
- LENGTH 函數
- OCTET_LENGTH
- SUBSTRING 函數
- FROM_HEX
- TO_HEX
- FROM_VARBYTE

- TO_VARBYTE
- GETBIT
- 載入 VARBYTE 資料類型的欄
- 卸載 VARBYTE 資料類型的欄

將 VARBYTE 資料類型與 Amazon Redshift 搭配使用時的限制

以下是將 VARBYTE 資料類型與 Amazon Redshift 搭配使用時的限制：

- Amazon Redshift Spectrum 僅支援 Parquet 和 ORC 檔案的 VARBYTE 資料類型。
- Amazon Redshift 查詢編輯器和 Amazon Redshift 查詢編輯器 v2 尚未完全支援 VARBYTE 資料類型。因此，在使用 VARBYTE 運算式時，請使用不同的 SQL 用戶端。

  作為使用查詢編輯器的解決方法，如果資料長度低於 64 KB 且內容是有效的 UTF-8，則可以將 VARBYTE 值轉換為 VARCHAR，例如：

  ```
  select to_varbyte('6162', 'hex')::varchar;
  ```
- 您不能將 VARBYTE 資料類別搭配 Python 或 Lambda 使用者定義函數 (UDF) 使用。
- 您無法從 VARBYTE 欄建立 HLLSKETCH 欄，也無法在 VARBYTE 欄上使用 APPROXIMATE COUNT DISTINCT。
- 大於 1 MB 的 VARBYTE 值只能從下列檔案格式中擷取：
  - Parquet
  - 文字
  - 逗號分隔值 (CSV)

類型相容性與轉換

接下來的內容將會說明在 Amazon Redshift 中，類型轉換規則與資料類型相容性的運作方式。

相容性

在資料庫各種操作的作業期間，會進行資料類型的比對，以及字面值與常數和資料類型的比對，包括下列的操作：

- 對資料表進行的資料處理語言 (DML) 操作
- UNION、INTERSECT 和 EXCEPT 查詢

- CASE 表達式
- 述詞的評估，例如 LIKE 和 IN
- 針對進行資料比較或擷取的 SQL 函式，進行評估
- 數學運算子的比較

這些操作的結果，取決於類型轉換規則和資料類型的相容性。兼容性意味著並不總是需要 one-to-one 匹配某個值和特定數據類型。由於某些資料類型是相容的，因此可進行隱含轉換或強制轉換 (如需詳細資訊，請參閱 隱含轉換類型)。當資料類型不相容時，有時您可以使用明確的轉換函式，來將值從一種資料類型轉換為另一種。

一般相容性與轉換規則

請注意下列的相容性與轉換規則：

- 一般而言，屬於相同類型類別的資料類型 (例如不同的數值資料類型)，彼此可以相容和隱含轉換。

  例如，進行隱含轉換時，您可以將小數值插入整數資料欄。小數會經過四捨五入而變成整數。或者，您可以從日期中擷取 2008 等數值，然後將該數值插入整數資料欄。
- 數值資料類型會強制執行嘗試插入 out-of-range 值時發生的溢位情況。例如，精確度為 5 的小數值，不符合精確度定義為 4 的小數資料欄。小數的整數或完整部分一律不會遭到截斷，但是小數的小數部分可以適當地向上或向下四捨五入。不過，從資料表所選取值的明確轉換結果，不會四捨五入。
- 不同類型的字元字串可以相容；VARCHAR 資料欄字串包含單位元組資料，CHAR 資料欄字串與其類似，可隱含轉換。包含多位元組資料的 VARCHAR 字串並不相容。此外，如果字串是適當的常值，則可以將字元字串轉換為日期、時間、時間戳記或數值；會忽略開頭或結尾的所有空格。相反地，您也可以將日期、時間、時間戳記或數值，轉換為固定長度或可變長度的字元字串。

  > **Note**
  >
  > 您想要轉換為數值類型的字元字串，必須包含表示數字的字元。例如，您可以將字串 '1.0' 或 '5.9' 轉換為小數值，但無法將字串 'ABC' 轉換為任何數值類型。

- 如果您將 DECIMAL 值與字元字串進行比較，Amazon Redshift 會嘗試將字元字串轉換為 DECIMAL 值。比較所有其他數值和字元字串時，數值會轉換為字元字串。若要強制執行相反的轉換 (例如，將字元字串轉換為整數，或將 DECIMAL 值轉換為字元字串)，請使用明確函數，例如 CAST。
- 若要將 64 位元的 DECIMAL 或 NUMERIC 值轉換為較高的精確度，您必須使用明確轉換函式，例如 CAST 或 CONVERT 函式。

- 將 DATE 或 TIMESTAMP 轉換為 TIMESTAMPTZ，或將 TIME 轉換為 TIMETZ 時，時區會設定為目前的工作階段時區。工作階段預設的時區為 UTC。如需設定工作階段時區的相關資訊，請參閱 timezone。
- 同樣地，TIMESTAMPTZ 轉換為 DATE、TIME 或 TIMESTAMP 時，也會使用目前工作階段的時區。工作階段預設的時區為 UTC。在轉換之後，會去掉時區的資訊。
- 用來表示指定時區之時間戳記的字元字串，會使用目前工作階段時區 (預設為 UTC) 轉換為 TIMESTAMPTZ。同樣地，代表指定時區之時間的字元字串，也會使用目前的工作階段時區 (預設為 UTC) 轉換為 TIMETZ。

隱含轉換類型

隱含轉換有兩種：

- 指派敘述中的隱含轉換，例如設定 INSERT 或 UPDATE 指令中的值。
- 表達式中的隱含轉換，例如在 WHERE 子句中進行比較。

下表列出了可以在指派敘述或表達式中隱含轉換的資料類型。您也可以使用明確轉換函式來進行這些轉換。

轉換前的類型	轉換後的類型
BIGINT (INT8)	BOOLEAN
	CHAR
	DECIMAL (NUMERIC)
	DOUBLE PRECISION (FLOAT8)
	INTEGER (INT、INT4)
	REAL (FLOAT4)
	SMALLINT (INT2)
	VARCHAR
CHAR	VARCHAR

轉換前的類型	轉換後的類型
DATE	CHAR
	VARCHAR
	TIMESTAMP
	TIMESTAMPTZ
DECIMAL (NUMERIC)	BIGINT (INT8)
	CHAR
	DOUBLE PRECISION (FLOAT8)
	INTEGER (INT、INT4)
	REAL (FLOAT4)
	SMALLINT (INT2)
	VARCHAR
DOUBLE PRECISION (FLOAT8)	BIGINT (INT8)
	CHAR
	DECIMAL (NUMERIC)
	INTEGER (INT、INT4)
	REAL (FLOAT4)
	SMALLINT (INT2)
	VARCHAR
INTEGER (INT、INT4)	BIGINT (INT8)
	BOOLEAN

轉換前的類型	轉換後的類型
	CHAR
	DECIMAL (NUMERIC)
	DOUBLE PRECISION (FLOAT8)
	REAL (FLOAT4)
	SMALLINT (INT2)
	VARCHAR
REAL (FLOAT4)	BIGINT (INT8)
	CHAR
	DECIMAL (NUMERIC)
	INTEGER (INT、INT4)
	SMALLINT (INT2)
	VARCHAR
SMALLINT (INT2)	BIGINT (INT8)
	BOOLEAN
	CHAR
	DECIMAL (NUMERIC)
	DOUBLE PRECISION (FLOAT8)
	INTEGER (INT、INT4)
	REAL (FLOAT4)
	VARCHAR

轉換前的類型	轉換後的類型
TIMESTAMP	CHAR
	DATE
	VARCHAR
	TIMESTAMPTZ
	TIME
TIMESTAMPTZ	CHAR
	DATE
	VARCHAR
	TIMESTAMP
	TIMETZ
TIME	VARCHAR
	TIMETZ
	INTERVAL DAY TO SECOND
TIMETZ	VARCHAR
	TIME
GEOMETRY	GEOGRAPHY
GEOGRAPHY	GEOMETRY

> **Note**
>
> TIMESTAMPTZ、TIMESTAMP、DATE、TIME、TIMETZ 或字元字串之間的隱含轉換，會使用目前工作階段的時區。關於設定目前的時區，詳細資訊請參閱 timezone。

GEOMETRY 和 GEOGRAPHY 資料類型無法隱含轉換為彼此以外的任何其他資料類型。如需詳細資訊，請參閱 CAST 函數。
VARBYTE 資料類型無法隱含轉換至任何其他資料類型。如需詳細資訊，請參閱 CAST 函數。

將動態類型用於 SUPER 資料類型

Amazon Redshift 會使用動態類型來處理無結構描述的 SUPER 資料，無需在查詢中使用資料類型之前先宣告資料類型。動態類型會使用導覽至 SUPER 資料欄的結果，而不必明確地將它們轉換為 Amazon Redshift 類型。如需將動態類型用於 SUPER 資料類型的相關資訊，請參閱動態類型。

您可以將 SUPER 值與其他資料類型互相轉換，但有一些例外情況。如需詳細資訊，請參閱 限制。

定序序列

Amazon Redshift 不支援地區特定或使用者定義的定序句。一般而言，如果缺少對資料值進行排序和比較時的地區特定規則，可能會影響到所有內容中任何述詞的結果。例如，ORDER BY 表達式和 MIN、MAX 及 RANK 等函式，會根據資料的二進位 UTF8 排序來傳回結果，並未將地區特定的字元納入考量。

表達式

主題

- 簡易表達式
- 複合運算式
- 表達式清單
- 賦值子查詢
- 函數表達式

表達式結合了一個或多個值、運算子或求值的函式。表達式的資料類型通常是其組成項目的資料類型。

簡易表達式

簡易表達式包括：

- 常數或常值
- 資料欄名稱或資料欄參考

- 純量函式
- 彙總 (集) 函式
- 視窗函式
- 賦值子查詢

簡易表達式的範例包括：

```
5+12
dateid
sales.qtysold * 100
sqrt (4)
max (qtysold)
(select max (qtysold) from sales)
```

複合運算式

複合表達式是由算術運算子聯結的一連串的簡單表達式。在複合表達式中使用的簡易表達式，必須傳回數值。

語法

```
expression
operator
expression | (compound_expression)
```

引數

運算式

求值的簡易表達式。

operator

複合的算術表達式可使用下列的運算子來建構，依此優先順序使用：

- ( )：用來控制求值順序的括號
- +、-：正號和負號/運算子
- ^、|/、||/：乘冪、平方根、立方根

- *、/、%：乘法、除法和模除運算子
- @：絕對值
- +、-：加和減
- &、|、#、~、<<、>>：AND、OR、NOT、向左移位、向右移位的位元運算子
- ||：串接

(compound_expression)

複合運算式可以使用括號來巢狀化。

範例

複合運算式的範例包括下列。

```
('SMITH' || 'JONES')
sum(x) / y
sqrt(256) * avg(column)
rank() over (order by qtysold) / 100
(select (pricepaid - commission) from sales where dateid = 1882) * (qtysold)
```

某些函式也可以嵌套於其他函式內。例如，任何純量函式都可以嵌套在另一個純量函式之內。下列的範例會傳回一組數字的絕對值總和：

```
sum(abs(qtysold))
```

視窗函式不能做為彙總函式或其他視窗函式的引數使用。下列的表達式會傳回錯誤：

```
avg(rank() over (order by qtysold))
```

視窗函式可以嵌套彙總函式。下列的表達式會求出值集合的總和，然後加以排序：

```
rank() over (order by sum(qtysold))
```

表達式清單

表達式清單是表達式的組合，可以出現在成員資格與比較條件 (WHERE 子句)，以及 GROUP BY 子句中。

語法

```
expression , expression , ... | (expression, expression, ...)
```

引數

運算式

求值的簡易表達式。表達式清單可以包含用英文逗號分隔的一個或多個表達式，或是用英文逗號分隔的一組或多組表達式。如果包含多組表達式，則每組都必須包含相同數量的表達式，並且用括號分隔。每組中的表達式數量，都必須符合條件中運算子前面的表達式數目。

範例

下列是條件中的表達式清單範例：

```
(1, 5, 10)
('THESE', 'ARE', 'STRINGS')
(('one', 'two', 'three'), ('blue', 'yellow', 'green'))
```

每組中的表達式數量，都必須符合陳述式前半部中的數目：

```
select * from venue
where (venuecity, venuestate) in (('Miami', 'FL'), ('Tampa', 'FL'))
order by venueid;

venueid |        venuename        | venuecity | venuestate | venueseats
---------+-------------------------+-----------+------------+------------
28 | American Airlines Arena | Miami     | FL         |          0
54 | St. Pete Times Forum    | Tampa     | FL         |          0
91 | Raymond James Stadium   | Tampa     | FL         |      65647
(3 rows)
```

賦值子查詢

賦值子查詢是括號中的正規 SELECT 查詢，只會傳回一個值 (一列和一個資料欄)。執行此查詢後，傳回的值會在外層的查詢中使用。如果子查詢傳回 0 列，則子查詢表達式的值為 null。如果子查詢傳回超過一列，Amazon Redshift 會傳回錯誤。子查詢可以參照父查詢傳來的變數，此變數會在子查詢的任何一次叫用期間中，做為常數使用。

您可以在叫用表達式的大多數陳述式中，使用賦值子查詢。賦值子查詢在下列的情況中並非有效的表達式：

- 做為表達式的預設值
- 在 GROUP BY 和 HAVING 子句中

範例

下列的子查詢會針對 2008 年一整年，計算出每次銷售平均支付的價格，外層的查詢接著會使用輸出中的此值，來比較每季的銷售平均價格：

```
select qtr, avg(pricepaid) as avg_saleprice_per_qtr,
(select avg(pricepaid)
from sales join date on sales.dateid=date.dateid
where year = 2008) as avg_saleprice_yearly
from sales join date on sales.dateid=date.dateid
where year = 2008
group by qtr
order by qtr;
qtr  | avg_saleprice_per_qtr | avg_saleprice_yearly
-------+-----------------------+----------------------
1     |                647.64 |               642.28
2     |                646.86 |               642.28
3     |                636.79 |               642.28
4     |                638.26 |               642.28
(4 rows)
```

函數表達式

語法

任何內建的函式皆可做為表達式使用。函式呼叫的語法，是函式的名稱後面接著括號中的引數列表。

```
function ( [expression [, expression...]] )
```

引數

函數

任何內建函式。如需一些範例函數，請參閱SQL 函數參考。

運算式

符合函式所預期資料類型和參數數目的任何表達式。

範例

```
abs (variable)
select avg (qtysold + 3) from sales;
select dateadd (day,30,caldate) as plus30days from date;
```

條件

主題

條件是一個或多個表達式和邏輯運算子 (判斷值為 true、false 或 unknown) 的陳述式。條件有時也稱為述詞。

> **Note**
>
> 所有的字串比較和 LIKE 模式比對，都會區分大小寫。例如，「A」和「a」不符。不過，您可以使用 ILIKE 述詞，來進行不區分大小寫的模式比對。

語法

```
comparison_condition
| logical_condition
| range_condition
```

```
| pattern_matching_condition
| null_condition
| EXISTS_condition
| IN_condition
```

比較條件

比較條件表示兩個值之間的邏輯關係。所有比較條件都是二元運算子，具有 Boolean 傳回類型。Amazon Redshift 支援下表中描述的比較運算子：

運算子	語法	描述
<	`a < b`	a 值小於 b 值。
>	`a > b`	a 值大於 b 值。
<=	`a <= b`	a 值小於或等於 b 值。
>=	`a >= b`	a 值大於或等於 b 值。
=	`a = b`	a 值等於 b 值。
<> 或 !=	`a <> b or a != b`	a 值不等於 b 值。
ANY | SOME	`a = ANY(subquery)`	a 值等於子查詢所傳回的任何值。
ALL	`a <> ALL or != ALL (subquery))`	a 值不等於子查詢所傳回的任何值。
IS TRUE | FALSE | UNKNOWN	`a IS TRUE`	a 值為布林值 TRUE。

使用須知

= ANY | SOME

ANY 和 SOME 關鍵字等同 IN 條件，如果有傳回一個或多個值的子查詢，其所傳回的值至少有一個在比較時為 true，則 ANY 和 SOME 會傳回 true。Amazon Redshift 只支援 ANY 和 SOME 的 = (等於) 條件。不支援不等式條件。

> **Note**
>
> 不支援 ALL 述詞。

<> ALL

ALL 關鍵字等同 NOT IN (請參閱 IN 條件 條件)，如果子查詢的結果中未包含運算式，將會傳回 true。Amazon Redshift 只支援 ALL 的 <> 或 != (不等於) 條件。不支援其他比較條件。

IS TRUE/FALSE/UNKNOWN

非 0 的值等於 TRUE、0 等於 FALSE，而 null 等於 UNKNOWN。請參閱 布林值 (Boolean) 類型 資料類型。

範例

下列是比較條件的一些簡單範例：

```
a = 5
a < b
min(x) >= 5
qtysold = any (select qtysold from sales where dateid = 1882
```

下列的查詢會從 VENUE 資料表傳回擁有超過 10,000 個座位的場地：

```
select venueid, venuename, venueseats from venue
where venueseats > 10000
order by venueseats desc;

venueid |           venuename            | venueseats
---------+--------------------------------+------------
83 | FedExField                    |      91704
 6 | New York Giants Stadium       |      80242
79 | Arrowhead Stadium             |      79451
78 | INVESCO Field                 |      76125
69 | Dolphin Stadium               |      74916
67 | Ralph Wilson Stadium          |      73967
76 | Jacksonville Municipal Stadium |      73800
89 | Bank of America Stadium       |      73298
72 | Cleveland Browns Stadium      |      73200
86 | Lambeau Field                 |      72922
```

```
...
(57 rows)
```

此範例會從 USERS 資料表中，選取喜歡搖滾樂的使用者 (USERID)：

```
select userid from users where likerock = 't' order by 1 limit 5;

userid
--------
3
5
6
13
16
(5 rows)
```

此範例會從 USERS 資料表中，選取不確定是否喜歡搖滾樂的使用者 (USERID)：

```
select firstname, lastname, likerock
from users
where likerock is unknown
order by userid limit 10;

firstname | lastname | likerock
----------+----------+----------
Rafael    | Taylor   |
Vladimir  | Humphrey |
Barry     | Roy      |
Tamekah   | Juarez   |
Mufutau   | Watkins  |
Naida     | Calderon |
Anika     | Huff     |
Bruce     | Beck     |
Mallory   | Farrell  |
Scarlett  | Mayer    |
(10 rows
```

具有 TIME 欄的範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME)，其中插入了三個值。

```
select time_val from time_test;
```

```
time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

下列範例會擷取每個 timetz_val 中的小時數。

```
select time_val from time_test where time_val < '3:00';
   time_val
---------------
 00:00:00.5550
 00:58:00
```

下列範例會比較兩個時間常值。

```
select time '18:25:33.123456' = time '18:25:33.123456';
 ?column?
----------
 t
```

具有 TIMTZ 欄的範例

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ)，其中插入了三個值。

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例只會選取小於 3:00:00 UTC 的 TIMETZ 值。將值轉換為 UTC 後進行比較。

```
select timetz_val from timetz_test where timetz_val < '3:00:00 UTC';

   timetz_val
---------------
 00:00:00.5550+00
```

下列範例會比較兩個 TIMETZ 常值。比較時會忽略時區。

```
select time '18:25:33.123456 PST' < time '19:25:33.123456 EST';

 ?column?
----------
 t
```

邏輯條件

邏輯條件會合併兩個條件的結果，來產生單一結果。所有的邏輯條件都是二元運算子，具有 Boolean 傳回類型。

語法

```
expression
{ AND | OR }
expression
NOT expression
```

邏輯條件使用三種值的布林邏輯，其中 null 值代表未知的關係。下表說明邏輯條件的結果，其中 E1 和 E2 表示表達式：

E1	E2	E1 AND E2	E1 OR E2	NOT E2
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE
TRUE	UNKNOWN (不明)	UNKNOWN (不明)	TRUE	UNKNOWN (不明)
FALSE	TRUE	FALSE	TRUE	
FALSE	FALSE	FALSE	FALSE	
FALSE	UNKNOWN (不明)	FALSE	UNKNOWN (不明)	
UNKNOWN (不明)	TRUE	UNKNOWN (不明)	TRUE	

E1	E2	E1 AND E2	E1 OR E2	NOT E2
UNKNOWN (不明)	FALSE	FALSE	UNKNOWN (不明)	
UNKNOWN (不明)	UNKNOWN (不明)	UNKNOWN (不明)	UNKNOWN (不明)	

NOT 運算子會在 AND 之前評估，而 AND 運算子會在 OR 運算子之前評估。如果使用任何括號，就可以覆蓋這個預設的評估順序。

範例

下列的範例會從 USERS 資料表，針對其中同時喜歡拉斯維加斯和運動的使用者，傳回其 USERID 和 USERNAME：

```
select userid, username from users
where likevegas = 1 and likesports = 1
order by userid;

userid | username
--------+----------
1 | JSG99FHE
67 | TWU10MZT
87 | DUF19VXU
92 | HYP36WEQ
109 | FPL38HZK
120 | DMJ24GUZ
123 | QZR22XGQ
130 | ZQC82ALK
133 | LBN45WCH
144 | UCX04JKN
165 | TEY68OEB
169 | AYQ83HGO
184 | TVX65AZX
...
(2128 rows)
```

下一個範例會從 USERS 資料表，針對其中喜歡拉斯維加斯或運動，或是這兩者的使用者，傳回其 USERID 和 USERNAME。此查詢會傳回前一個範例的所有輸出資料，加上只喜歡拉斯維加斯或運動的使用者。

```
select userid, username from users
where likevegas = 1 or likesports = 1
order by userid;

userid | username
--------+----------
1 | JSG99FHE
2 | PGL08LJI
3 | IFT66TXU
5 | AEB55QTM
6 | NDQ15VBM
9 | MSD36KVR
10 | WKW41AIW
13 | QTF33MCG
15 | OWU78MTR
16 | ZMG93CDD
22 | RHT62AGI
27 | KOY02CVE
29 | HUH27PKK
...
(18968 rows)
```

下列的查詢在 OR 條件周圍加上括號，以找出在紐約或加州上演「馬克白」的場地：

```
select distinct venuename, venuecity
from venue join event on venue.venueid=event.venueid
where (venuestate = 'NY' or venuestate = 'CA') and eventname='Macbeth'
order by 2,1;

venuename                |   venuecity
-----------------------------------------+---------------
Geffen Playhouse                         | Los Angeles
Greek Theatre                            | Los Angeles
Royce Hall                               | Los Angeles
American Airlines Theatre                | New York City
August Wilson Theatre                    | New York City
Belasco Theatre                          | New York City
Bernard B. Jacobs Theatre                | New York City
...
```

如果移除此範例中的括號，將會改變查詢的邏輯和結果。

下列的範例使用 NOT 運算子：

```
select * from category
where not catid=1
order by 1;

catid | catgroup |  catname  |                  catdesc
-------+----------+-----------+--------------------------------------------
2 | Sports   | NHL       | National Hockey League
3 | Sports   | NFL       | National Football League
4 | Sports   | NBA       | National Basketball Association
5 | Sports   | MLS       | Major League Soccer
...
```

下列範例使用 NOT 條件，後接 AND 條件：

```
select * from category
where (not catid=1) and catgroup='Sports'
order by catid;

catid | catgroup | catname |             catdesc
-------+----------+---------+---------------------------------
2 | Sports   | NHL     | National Hockey League
3 | Sports   | NFL     | National Football League
4 | Sports   | NBA     | National Basketball Association
5 | Sports   | MLS     | Major League Soccer
(4 rows)
```

模式比對條件

主題

- LIKE
- SIMILAR TO
- POSIX 運算子

模式比對運算子會依據條件表達式中所指定的模式，來搜尋字串，並根據是否找到符合的結果，傳回 true 或 false。Amazon Redshift 使用三種方法進行模式比對：

- LIKE 表達式

LIKE 運算子會利用模式 (此模式使用萬用字元 % (百分比) 和 _ (底線)) 來比較字串表達式 (例如資料欄的名稱)。LIKE 模式比對的範圍一律涵蓋整個字串。LIKE 會進行區分大小寫的比對，ILIKE 則會進行不區分大小寫的比對。

- SIMILAR TO 規則表達式

  SIMILAR TO 運算子會利用 SQL 標準規則表達式的模式，來比對字串表達式，此模式包含一組模式比對中繼字元，其中包括 LIKE 運算子所支援的兩個字元。SIMILAR TO 會比對整個字串，並進行區分大小寫的比對。

- POSIX 樣式規則表達式

  POSIX 規則表達式提供了更強大的方法，來進行 LIKE 和 SIMILAR TO 運算子以外的模式比對。POSIX 規則表達式可比對字串的任何部分，並進行區分大小寫的比對。

使用 SIMILAR TO 或 POSIX 運算子進行的規則表達式比對，其運算成本非常昂貴。我們建議盡可能使用 LIKE，尤其是在處理極為龐大的列數時。例如，下列的查詢在功能上相同，但相較於使用規則運算式的查詢，使用 LIKE 的查詢，其執行速度快上好幾倍：

```
select count(*) from event where eventname SIMILAR TO '%(Ring|Die)%';
select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die%';
```

LIKE

LIKE 運算子會利用模式 (此模式使用萬用字元 % (百分比) 和 _ (底線)) 來比較字串表達式 (例如資料欄的名稱)。LIKE 模式比對的範圍一律涵蓋整個字串。若要比對字串中任意位置的序列，模式必須以百分比符號開頭和結尾。

LIKE 區分大小寫；ILIKE 不區分大小寫。

語法

```
expression [ NOT ] LIKE | ILIKE pattern [ ESCAPE 'escape_char' ]
```

引數

運算式

有效的 UTF-8 字元表達式，例如資料欄的名稱。

LIKE | ILIKE

LIKE 會進行區分大小寫的模式比對。ILIKE 會針對單位元組 UTF-8 (ASCII) 字元，進行不區分大小寫的模式比對。若要對多位元組字元執行不區分大小寫的模式比對，請在具有 LIKE 條件的 expression 和 pattern 上使用 LOWER 函數。

不同於比較述詞，例如 = 和 <>，LIKE 和 ILIKE 述詞並未隱含忽略結尾空格。若要忽略結尾空格，請 RTRIM 或將 CHAR 資料欄明確轉換為 VARCHAR。

~~ 運算子相當於 LIKE，而 ~~* 相當於 ILIKE。此外，!~~ 和 !~~* 運算子相當於 NOT LIKE 和 NOT ILIKE。

pattern

有效的 UTF-8 字元表達式，包含要比對的模式。

escape_char

字元表達式，將會用來逸出模式中的中繼字元。預設值為兩個反斜線 (「\\」)。

如果 pattern 未包含任何中繼字元，則模式只代表字串本身，此時 LIKE 的功用如同等於運算子。

兩個字元表達式都可以是 CHAR 或 VARCHAR 資料類型。如果不同，Amazon Redshift 會將 pattern 轉換為 expression 的資料類型。

LIKE 支援下列的模式比對中繼字元：

運算子	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。

範例

下表顯示範例，示範使用 LIKE 進行的模式比對：

表達式	傳回值
`'abc' LIKE 'abc'`	True

表達式	傳回值
'abc' LIKE 'a%'	True
'abc' LIKE '_B_'	False
'abc' ILIKE '_B_'	True
'abc' LIKE 'c%'	False

下列範例會找出名稱以「E」開頭的所有城市：

```
select distinct city from users
where city like 'E%' order by city;
city
---------------
East Hartford
East Lansing
East Rutherford
East St. Louis
Easthampton
Easton
Eatontown
Eau Claire
...
```

下列範例會找出姓氏中包含「ten」的使用者：

```
select distinct lastname from users
where lastname like '%ten%' order by lastname;
lastname
-------------
Christensen
Wooten
...
```

以下範例示範如何比對多個模式。

```
select distinct lastname from tickit.users
where lastname like 'Chris%' or lastname like '%Wooten' order by lastname;
lastname
```

```
-------------
Christensen
Christian
Wooten
...
```

下列範例會找出名稱中第 3 個和第 4 個字元為「ea」的城市。此指令使用 ILIKE 來示範不區分大小寫：

```
select distinct city from users where city ilike '__EA%' order by city;
city
-------------
Brea
Clearwater
Great Falls
Ocean City
Olean
Wheaton
(6 rows)
```

下列的範例使用預設的逸出字串 (\\)，來搜尋包含「_」的字串 (文字 start 後跟底線 _)：

```
select tablename, "column" from pg_table_def
where "column" like '%start\\_%'
limit 5;

     tablename     |    column
-------------------+---------------
 stl_s3client      | start_time
 stl_tr_conflict   | xact_start_ts
 stl_undone        | undo_start_ts
 stl_unload_log    | start_time
 stl_vacuum_detail | start_row
(5 rows)
```

下列的範例將「^」指定為逸出字元，然後使用該逸出字元來搜尋包含「_」的字串 (文字 start 後跟底線 _)：

```
select tablename, "column" from pg_table_def
where "column" like '%start^_%' escape '^'
limit 5;
```

```
     tablename     |    column
-------------------+---------------
 stl_s3client      | start_time
 stl_tr_conflict   | xact_start_ts
 stl_undone        | undo_start_ts
 stl_unload_log    | start_time
 stl_vacuum_detail | start_row
(5 rows)
```

下列範例會使用 ~~* 運算子執行不區分大小寫 (ILIKE)，搜尋以「Ag」開頭的城市。

```
select distinct city from users where city ~~* 'Ag%' order by city;

city
------------
Agat
Agawam
Agoura Hills
Aguadilla
```

SIMILAR TO

SIMILAR TO 運算子會利用 SQL 標準規則表達式的模式，來比對字串表達式 (例如資料欄的名稱)。SQL 標準規則表達式的模式可包含一組模式比對中繼字元，其中包括 LIKE 運算子所支援的兩個字元。

SIMILAR TO 運算子只會在其模式符合整個字串時傳回 true，不像 POSIX 規則表達式，其模式可以符合字串的任何部分。

SIMILAR TO 會進行區分大小寫的比對。

> **Note**
>
> 使用 SIMILAR TO 進行的規則表達式比對，其運算成本非常昂貴。我們建議盡可能使用 LIKE，尤其是在處理極為龐大的列數時。例如，下列的查詢在功能上相同，但相較於使用規則運算式的查詢，使用 LIKE 的查詢，其執行速度快上好幾倍：
>
> ```
> select count(*) from event where eventname SIMILAR TO '%(Ring|Die)%';
> select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die
> %';
> ```

語法

```
expression [ NOT ] SIMILAR TO pattern [ ESCAPE 'escape_char' ]
```

引數

運算式

有效的 UTF-8 字元表達式，例如資料欄的名稱。

SIMILAR TO

SIMILAR TO 會針對 expression 中的整個字串，進行區分大小寫的模式比對。

pattern

有效的 UTF-8 字元表達式，代表 SQL 標準規則表達式的模式。

escape_char

字元表達式，將會用來逸出模式中的中繼字元。預設值為兩個反斜線 (「\\」)。

如果模式未包含任何中繼字元，則模式只代表字串本身。

兩個字元表達式都可以是 CHAR 或 VARCHAR 資料類型。如果不同，Amazon Redshift 會將 pattern 轉換為 expression 的資料類型。

SIMILAR TO 支援下列的模式比對中繼字元：

運算子	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。
|	表示交替 (兩種替代選項中的任一種)。
*	重複前一個項目 0 次或更多次。
+	重複前一個項目 1 次或更多次。
?	重複前一個項目 0 次或 1 次。
{m}	重複前一個項目 m 次。

運算子	描述
{m,}	重複前一個項目 m 次或更多次。
{m,n}	重複前一個項目至少 m 次，而且不超過 n 次。
()	括號可將多個項目分組為單一邏輯項目。
[...]	方括號表達式表示字元類別，如同在 POSIX 規則表達式中的表示分式。

範例

下表顯示範例，示範使用 SIMILAR TO 進行的模式比對：

表達式	傳回值
'abc' SIMILAR TO 'abc'	True
'abc' SIMILAR TO '_b_'	True
'abc' SIMILAR TO '_A_'	False
'abc' SIMILAR TO '%(b|d)%'	True
'abc' SIMILAR TO '(b|c)%'	False
'AbcAbcdefgefg12efgefg12' SIMILAR TO '((Ab)?c)+d((efg)+(12))+'	True
'aaaaaab11111xy' SIMILAR TO 'a{6}_[0-9]{5}(x|y){2}'	True
'$0.87' SIMILAR TO '$[0-9]+(.[0-9][0-9])?'	True

下列範例會找出名稱中包含「E」或「H」的城市：

```
SELECT DISTINCT city FROM users
WHERE city SIMILAR TO '%E%|%H%' ORDER BY city LIMIT 5;
```

```
      city
-----------------
 Agoura Hills
 Auburn Hills
 Benton Harbor
 Beverly Hills
 Chicago Heights
```

下列的範例使用預設的逸出字串 (「\\」)，來搜尋包含「_」的字串：

```
SELECT tablename, "column" FROM pg_table_def
WHERE "column" SIMILAR TO '%start\\_%'
ORDER BY tablename, "column" LIMIT 5;

        tablename         |       column
--------------------------+---------------------
 stcs_abort_idle          | idle_start_time
 stcs_abort_idle          | txn_start_time
 stcs_analyze_compression | start_time
 stcs_auto_worker_levels  | start_level
 stcs_auto_worker_levels  | start_wlm_occupancy
```

下列的範例將「^」指定為逸出字元，然後使用該逸出字元來搜尋包含「_」的字串：

```
SELECT tablename, "column" FROM pg_table_def
WHERE "column" SIMILAR TO '%start^_%' ESCAPE '^'
ORDER BY tablename, "column" LIMIT 5;

        tablename         |       column
--------------------------+---------------------
 stcs_abort_idle          | idle_start_time
 stcs_abort_idle          | txn_start_time
 stcs_analyze_compression | start_time
 stcs_auto_worker_levels  | start_level
 stcs_auto_worker_levels  | start_wlm_occupancy
```

POSIX 運算子

POSIX 規則運算式是指定比對模式的字元序列。如果字串是規則運算式所描述之規則集的成員，就會比對規則運算式。

POSIX 規則表達式提供了更強大的方法，來進行 LIKE 和 SIMILAR TO 運算子以外的模式比對。POSIX 規則表達式的模式可以符合字串的任何部分，不像 SIMILAR TO 運算子只在其模式符合整個字串時，才會傳回 true。

> **Note**
>
> 使用或 POSIX 運算子進行的規則表達式比對，其運算成本非常昂貴。我們建議盡可能使用 LIKE，尤其是在處理極為龐大的列數時。例如，下列的查詢在功能上相同，但相較於使用規則運算式的查詢，使用 LIKE 的查詢，其執行速度快上好幾倍：
>
> ```
> select count(*) from event where eventname ~ '.*(Ring|Die).*';
> select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die
> %';
> ```

語法

```
expression [ ! ] ~ pattern
```

引數

運算式

有效的 UTF-8 字元表達式，例如資料欄的名稱。

!

否定運算子。不符合規則運算式。

~

針對 expression 的任何子字串，進行區分大小寫的比對。

> **Note**
>
> ~~ 是 LIKE 的同義詞。

pattern

代表規則運算式模式的字串常值。

如果模式未包含萬用字元，則模式只代表字串本身。

若要搜尋包含中繼字元 (例如「. * | ? 」等) 的字串，請使用兩個反斜線 (「 \\」) 來逸出字元。不像 SIMILAR TO 和 LIKE，POSIX 規則表達式的語法不支援使用者定義的逸出字元。

兩個字元表達式都可以是 CHAR 或 VARCHAR 資料類型。如果不同，Amazon Redshift 會將 pattern 轉換為 expression 的資料類型。

所有的字元表達式都可以是 CHAR 或 VARCHAR 資料類型。如果運算式的資料類型不同，Amazon Redshift 會將其轉換為 expression 的資料類型。

POSIX 模式比對支援下列的中繼字元：

POSIX	描述
.	比對任一個單一字元。
*	比對出現 0 次或更多次。
+	比對出現 1 次或更多次。
?	比對出現 0 次或 1 次。
|	指定交替比對；例如，E | H 代表 E 或 H。
^	符合 beginning-of-line 字元。
$	符合 end-of-line 字元。
$	比對字串結尾。
[]	方括號用來表示符合的清單，應符合清單中的一個表達式。插入符號 (^) 會出現在不符合清單的前面，此清單會比對除了清單中所顯示表達式以外的所有字元。
( )	括號可將多個項目分組為單一邏輯項目。
{m}	重複前一個項目 m 次。
{m,}	重複前一個項目 m 次或更多次。
{m,n}	重複前一個項目至少 m 次，而且不超過 n 次。

POSIX	描述
[: :]	比對 POSIX 字元類別內的任何字元。在下列的字元類別中，Amazon Redshift 只支援 ASCII 字元：[:alnum:] 、[:alpha:] 、[:lower:] 、[:upper:]

Amazon Redshift 支援下列的 POSIX 字元類別。

字元類別	描述
[[:alnum:]]	所有 ASCII 英數字元
[[:alpha:]]	所有 ASCII 字母字元
[[:blank:]]	所有空白字元
[[:cntrl:]]	所有控制字元 (非列印)
[[:digit:]]	所有數字位數
[[:lower:]]	所有小寫 ASCII 字母字元
[[:punct:]]	所有標點符號字元
[[:space:]]	所有空格字元 (非列印)
[[:upper:]]	所有大寫 ASCII 字母字元
[[:xdigit:]]	所有的有效十六進位字元

Amazon Redshift 在規則運算式中支援下列受到 Perl 影響的運算子。使用兩個反斜線 (「\\」) 來逸出運算子。

運算子	描述	同等的字元類別表達式
\\d	數字字元	[[:digit:]]
\\D	非數字字元	[^[:digit:]]

運算子	描述	同等的字元類別表達式
\\w	字詞字元	[[:word:]]
\\W	非字詞字元	[^[:word:]]
\\s	空白字元	[[:space:]]
\\S	非空白字元	[^[:space:]]
\\b	邊界字	

範例

下表顯示範例，示範使用 POSIX 運算子進行的模式比對：

表達式	傳回值
'abc' ~ 'abc'	True
'abc' ~ 'a'	True
'abc' ~ 'A'	False
'abc' ~ '.*(b|d).*'	True
'abc' ~ '(b|c).*'	True
'AbcAbcdefgefg12efgefg12' ~ '((Ab)?c)+d((efg)+(12))+'	True
'aaaaaab11111xy' ~ 'a{6}.[1]{5}(x|y){2}'	True
'$0.87' ~ '\\$[0-9]+(\\.[0-9][0-9])?'	True
'ab c' ~ '[[:space:]]'	True
'ab c' ~ '\\s'	True

表達式	傳回值
' ' ~ '\\S'	False

下列範例會找出名稱中包含 E 或 H 的城市：

```
SELECT DISTINCT city FROM users
WHERE city ~ '.*E.*|.*H.*' ORDER BY city LIMIT 5;

      city
-----------------
 Agoura Hills
 Auburn Hills
 Benton Harbor
 Beverly Hills
 Chicago Heights
```

下列範例會找出名稱中不包含 E 或 H 的所有城市：

```
SELECT DISTINCT city FROM users WHERE city !~ '.*E.*|.*H.*' ORDER BY city LIMIT 5;

      city
-----------------
 Aberdeen
 Abilene
 Ada
 Agat
 Agawam
```

下列的範例使用逸出字串 (「\\」) 來搜尋包含英文句點的字串。

```
SELECT venuename FROM venue
WHERE venuename ~ '.*\\..*'
ORDER BY venueid;

          venuename
------------------------------
 St. Pete Times Forum
 Jobing.com Arena
 Hubert H. Humphrey Metrodome
 U.S. Cellular Field
```

```
 Superpages.com Center
 E.J. Nutter Center
 Bernard B. Jacobs Theatre
 St. James Theatre
```

BETWEEN 範圍條件

BETWEEN 條件會使用關鍵字 BETWEEN 和 AND，來檢定表達式是否在一系列值的範圍內。

語法

```
expression [ NOT ] BETWEEN expression AND expression
```

表達式可以是數值、字元或日期時間 (datetime) 資料類型，但這些類型必須相容。範圍包含端點。

範例

第一個範例會計算有多少交易已登錄售出 2、3 或 4 張票券：

```
select count(*) from sales
where qtysold between 2 and 4;

count
--------
104021
(1 row)
```

範圍條件包含開頭值與結尾值。

```
select min(dateid), max(dateid) from sales
where dateid between 1900 and 1910;

min  | max
-----+-----
1900 | 1910
```

範圍條件中第一個表達式的值，必須小於第二個表達式的值。由於表達式的值，下列的範例一律會傳回 0 列：

```
select count(*) from sales
```

```
where qtysold between 4 and 2;

count
-------
0
(1 row)
```

不過，套用 NOT 修飾符將會反轉邏輯，產生所有列的計數：

```
select count(*) from sales
where qtysold not between 4 and 2;

count
--------
172456
(1 row)
```

下列的查詢會傳回擁有 20,000 到 50,000 個座位的場地清單：

```
select venueid, venuename, venueseats from venue
where venueseats between 20000 and 50000
order by venueseats desc;

venueid |       venuename               | venueseats
---------+-------------------------------+------------
116 | Busch Stadium                   |      49660
106 | Rangers BallPark in Arlington |      49115
96 | Oriole Park at Camden Yards   |      48876
...
(22 rows)
```

下列範例示範使用 BETWEEN 的日期值：

```
select salesid, qtysold, pricepaid, commission, saletime
from sales
where eventid between 1000 and 2000
   and saletime between '2008-01-01' and '2008-01-03'
order by saletime asc;

salesid | qtysold | pricepaid | commission |   saletime
--------+---------+-----------+------------+---------------
  65082 |       4 |       472 |       70.8 | 1/1/2008 06:06
```

```
 110917 |       1 |       337 |      50.55 | 1/1/2008 07:05
 112103 |       1 |       241 |      36.15 | 1/2/2008 03:15
 137882 |       3 |      1473 |     220.95 | 1/2/2008 05:18
  40331 |       2 |        58 |        8.7 | 1/2/2008 05:57
 110918 |       3 |      1011 |     151.65 | 1/2/2008 07:17
  96274 |       1 |       104 |       15.6 | 1/2/2008 07:18
 150499 |       3 |       135 |      20.25 | 1/2/2008 07:20
  68413 |       2 |       158 |       23.7 | 1/2/2008 08:12
```

請注意，儘管 BETWEEN 的範圍包含在內，但日期預設的時間值為 00:00:00。範例查詢的唯一有效 1 月 3 日列是銷售時間為 1/3/2008 00:00:00 的列。

Null 條件

null 條件會檢定 null (缺少值或是值未知)。

語法

```
expression IS [ NOT ] NULL
```

引數

運算式

任何表達式，例如資料欄。

IS NULL

表達式的值如果是 null 則為 true，表達式的值如果包含值，則為 false。

IS NOT NULL

表達式的值如果是 null 則為 false，表達式的值如果包含值，則為 true。

範例

此範例顯示 SALES 資料表的 QTYSOLD 欄位中有多少次包含 null：

```
select count(*) from sales
where qtysold is null;
count
-------
0
```

```
(1 row)
```

EXISTS 條件

EXISTS 條件會檢定在子查詢中是否存在列，如果子查詢傳回至少一列，則傳回 true。如果指定 NOT，則此條件會在子查詢未傳回任何列時傳回 true。

語法

```
[ NOT ] EXISTS (table_subquery)
```

引數

EXISTS

當 table_subquery 傳回至少一列時，其值為 true。

NOT EXISTS

當 table_subquery 未傳回任何列時，其值為 true。

table_subquery

子查詢，會評估包含一個或多個欄和一列或多列的資料表。

範例

此範例會針對具有任何類型銷售的日期，傳回所有的日期識別碼，一次一個：

```
select dateid from date
where exists (
select 1 from sales
where date.dateid = sales.dateid
)
order by dateid;

dateid
--------
1827
1828
1829
...
```

IN 條件

IN 條件會檢定值是否隸屬於一組值或子查詢。

語法

```
expression [ NOT ] IN (expr_list | table_subquery)
```

引數

運算式

數值、字元或日期時間 (datetime) 表達式，會根據 expr_list 或 table_subquery 進行評估，而且必須與該清單或子查詢的資料類型相容。

expr_list

用英文逗號分隔的一個或多個表達式，或是用英文逗號分隔的一組或多組表達式 (用括號括住)。

table_subquery

子查詢，會評估包含一列或多列的資料表，但是其選擇清單中只限包含一個欄。

IN | NOT IN

如果表達式是表達式清單或查詢的成員，IN 會傳回 true。如果表達式不是成員，NOT IN 會傳回 true。在下列情況中，IN 和 NOT IN 會傳回 Null，而且不會傳回任何列：如果 expression 產生 null；或如果沒有符合的 expr_list 或 table_subquery 值，而且這些比較列其中至少有一列產生 null。

範例

只有這些列出的值，才會讓下列條件傳回 true：

```
qtysold in (2, 4, 5)
date.day in ('Mon', 'Tues')
date.month not in ('Oct', 'Nov', 'Dec')
```

大型 IN 清單的最佳化

為了實現最佳化的查詢效能，包含超過 10 個值的 IN 清單，會在內部轉換為純量陣列。包含不到 10 個值的 IN 清單，會轉換為一系列的 OR 述詞。支援

這個最佳化功能的包括 SMALLINT、INTEGER、BIGINT、REAL、DOUBLE PRECISION、BOOLEAN、CHAR、VARCHAR、DATE、TIMESTAMP 和 TIMESTAMPTZ 等資料類型。

請檢視查詢的 EXPLAIN 輸出，以查看這個最佳化機制的效果。例如：

```
explain select * from sales
QUERY PLAN
------------------------------------------------------------------------
XN Seq Scan on sales  (cost=0.00..6035.96 rows=86228 width=53)
Filter: (salesid = ANY ('{1,2,3,4,5,6,7,8,9,10,11}'::integer[]))
(2 rows)
```

SQL 命令

SQL 語言包含您用來建立和操作資料庫物件、執行查詢、載入資料表，以及修改資料表中資料的命令。

Amazon Redshift 以 PostgreSQL 為基礎。在設計和開發您的資料倉儲應用程式時，您必須知道 Amazon Redshift 與 PostgreSQL 之間有多項重要的差異。如需 Amazon Redshift SQL 與 PostgreSQL 之間差異的相關資訊，請參閱 Amazon Redshift 和 PostgreSQL。

> **Note**
>
> 單一 SQL 陳述式的大小上限為 16 MB。

主題

- ABORT
- ALTER DATABASE
- ALTER DATASHARE
- ALTER DEFAULT PRIVILEGES
- ALTER EXTERNAL VIEW (預覽)
- ALTER FUNCTION
- ALTER GROUP
- ALTER IDENTITY PROVIDER

- ALTER MASKING POLICY
- ALTER MATERIALIZED VIEW
- ALTER RLS POLICY
- ALTER ROLE
- ALTER PROCEDURE
- ALTER SCHEMA
- ALTER SYSTEM
- ALTER TABLE
- ALTER TABLE APPEND
- ALTER USER
- ANALYZE
- ANALYZE COMPRESSION
- ATTACH MASKING POLICY
- ATTACH RLS POLICY
- BEGIN
- CALL
- 取消
- CLOSE
- COMMENT
- COMMIT
- COPY
- CREATE DATABASE
- CREATE DATASHARE
- CREATE EXTERNAL FUNCTION
- CREATE EXTERNAL SCHEMA
- CREATE EXTERNAL TABLE
- CREATE EXTERNAL VIEW (預覽)
- CREATE FUNCTION
- CREATE GROUP
- CREATE IDENTITY PROVIDER

- CREATE LIBRARY
- CREATE MASKING POLICY
- CREATE MATERIALIZED VIEW
- CREATE MODEL
- CREATE PROCEDURE
- CREATE RLS POLICY
- CREATE ROLE
- CREATE SCHEMA
- CREATE TABLE
- CREATE TABLE AS
- CREATE USER
- CREATE VIEW
- DEALLOCATE
- DECLARE
- DELETE
- DESC DATASHARE
- DESC IDENTITY PROVIDER
- DETACH MASKING POLICY
- DETACH RLS POLICY
- DROP DATABASE
- DROP DATASHARE
- DROP EXTERNAL VIEW (預覽)
- DROP FUNCTION
- DROP GROUP
- DROP IDENTITY PROVIDER
- DROP LIBRARY
- DROP MASKING POLICY
- DROP MODEL
- DROP MATERIALIZED VIEW
- DROP PROCEDURE

- DROP RLS POLICY
- DROP ROLE
- DROP SCHEMA
- DROP TABLE
- DROP USER
- DROP VIEW
- 結束
- EXECUTE
- EXPLAIN
- FETCH
- GRANT
- INSERT
- INSERT (外部資料表)
- LOCK
- MERGE
- PREPARE
- REFRESH MATERIALIZED VIEW
- RESET
- REVOKE
- ROLLBACK
- SELECT
- SELECT INTO
- SET
- SET SESSION AUTHORIZATION
- SET SESSION CHARACTERISTICS
- SHOW
- SHOW COLUMNS
- SHOW EXTERNAL TABLE
- SHOW DATABASES
- SHOW MODEL

- SHOW DATASHARES
- SHOW PROCEDURE
- SHOW SCHEMAS
- SHOW TABLE
- SHOW TABLES
- SHOW VIEW
- START TRANSACTION
- TRUNCATE
- UNLOAD
- UPDATE
- VACUUM

ABORT

停止目前執行的交易，並捨棄該交易所做的所有更新。ABORT 不影響已完成的交易。

此命令會執行與 ROLLBACK 命令相同的功能。如需相關資訊，請參閱ROLLBACK。

語法

```
ABORT [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。

TRANSACTION

選用的關鍵字；WORK 和 TRANSACTION 為同義詞。

範例

下列範例會建立資料表，然後開始交易，將資料插入資料表中。接著 ABORT 命令就會轉返資料插入操作，而使資料表變成空白。

以下命令會建立名為 MOVIE_GROSS 的範例資料表：

```
create table movie_gross( name varchar(30), gross bigint );
```

下一個命令集會開始交易，將兩個資料列插入資料表中：

```
begin;

insert into movie_gross values ( 'Raiders of the Lost Ark', 23400000);

insert into movie_gross values ( 'Star Wars', 10000000 );
```

接著，以下命令會從資料表中選取資料，表示該資料已成功插入：

```
select * from movie_gross;
```

命令輸出會顯示這兩個資料列都已成功插入：

```
         name           |  gross
------------------------+----------
Raiders of the Lost Ark | 23400000
Star Wars               | 10000000
(2 rows)
```

現在這個命令會將資料變更轉返為交易開始時的狀態：

```
abort;
```

從資料表選取資料現在會顯示空白資料表：

```
select * from movie_gross;

 name | gross
------+-------
(0 rows)
```

ALTER DATABASE

變更資料庫的屬性。

所需權限

若要使用 ALTER DATABASE，需要下列其中一項權限。

- 超級使用者
- 具有 ALTER DATABASE 權限的使用者
- 資料庫擁有者

語法

```
ALTER DATABASE database_name
{ RENAME TO new_name
| OWNER TO new_owner
| CONNECTION LIMIT { limit | UNLIMITED }
| COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }
| ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT }
| INTEGRATION REFRESH {{ ALL | INERROR } TABLES [IN SCHEMA schema [, ...]] |
 TABLE schema.table [, ...]}
}
```

參數

database_name

要修改的資料庫名稱。通常您會修改目前未連接的資料庫；無論何種情況下，變更都只會在後續工作階段中生效。您可以變更目前資料庫的擁有者，但無法將它重新命名：

```
alter database tickit rename to newtickit;
ERROR:  current database may not be renamed
```

RENAME TO

重新命名指定的資料庫。如需有效名稱的相關資訊，請參閱 名稱與識別碼。您無法重新命名 dev、padb_harvest、template0 或 template1 或 sys:internal 資料庫，也無法重新命名目前的資料庫。只有資料庫擁有者或 superuser (p. 760) 能夠重新命名資料庫；非超級使用者的擁有者必須同時具備 CREATEDB 權限。

new_name

新的資料庫名稱。

OWNER TO

變更所指定資料庫的擁有者。您可以變更目前資料庫或其他資料庫的擁有者。但只有超級使用者可以變更擁有者。

new_owner

新的資料庫擁有者。新的擁有者必須是具備寫入權限的現有資料庫使用者。如需使用者權限的相關資訊，請參閱 GRANT。

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字可允許同時連線的最大數目。另外也可能限制每位使用者的連線數目。如需詳細資訊，請參閱 CREATE USER。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 STV_SESSIONS 系統畫面。

> **Note**
>
> 如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }

指定字串搜尋或比較是否區分大小寫的子句。

您可以變更目前空資料庫的區分大小寫設定。

您必須擁有目前資料庫的權限，才能變更區分大小寫的設定。具有 CREATE DATABASE 權限的超級使用者或資料庫擁有者也可以變更資料庫的區分大小寫設定。

ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT }

指定對資料庫執行查詢時所用隔離層級的子句。

- SERIALIZABLE 隔離 — 為並行交易提供完整的序列化。如需詳細資訊，請參閱 可序列化隔離。
- SNAPSHOT 隔離 — 提供隔離層級，防止更新和刪除衝突。

如需隔離層級的相關資訊，請參閱 CREATE DATABASE。

修改資料庫的隔離層級時，請考慮下列項目：

- 您必須擁有目前資料庫的超級使用者或 CREATE DATABASE 權限，才能變更資料庫隔離層級。

- 您無法變更 dev 資料庫的隔離層級。
- 您無法變更交易區塊中的隔離等級。
- 如果其他使用者連線到資料庫，修改隔離層級命令就會失敗。
- 修改隔離層級命令可以修改目前工作階段的隔離層級設定。

INTEGRATION REFRESH {{ ALL | INERROR } TABLES [IN SCHEMA schema [, ...]] | TABLE schema.table [, ...]}

指定 Amazon Redshift 是否會重新整理指定結構描述或資料表中的所有資料表或發生錯誤之資料表的子句。重新整理會觸發指定結構描述或資料表中的資料表，從來源資料庫完全複製。

如需詳細資訊，請參閱 Amazon Redshift 管理指南中的使用零 ETL 整合。如需整合狀態的詳細資訊，請參閱 SVV_INTEGRATION_TABLE_STATE 和 SVV_INTEGRATION。

使用須知

ALTER DATABASE 命令會套用至後續工作階段，而非目前工作階段。您必須重新連線到修改過的資料庫，才會看見變更的效果。

範例

以下範例會將名為 TICKIT_SANDBOX 的資料庫重新命名為 TICKIT_TEST：

```
alter database tickit_sandbox rename to tickit_test;
```

下列範例會將 TICKIT 資料庫 (目前資料庫) 的擁有者變更為 DWUSER：

```
alter database tickit owner to dwuser;
```

下列範例會變更 sampledb 資料庫的資料庫大小寫設定：

```
ALTER DATABASE sampledb COLLATE CASE_INSENSITIVE;
```

下列範例會修改名稱為 **sampledb** 且具有 SNAPSHOT 隔離層級的資料庫。

```
ALTER DATABASE sampledb ISOLATION LEVEL SNAPSHOT;
```

下列範例會重新整理零 ETL 整合之 **sample_integration_db** 中的資料表 **sample_table1** 和 **sample_table2** 資料庫。

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH TABLES sample_table1,
 sample_table2;
```

下列範例會重新整理零 ETL 整合中所有已同步處理和失敗的資料表。

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH ALL tables;
```

下列範例會重新整理結構描述 **sample_schema** 中 ErrorState 的所有資料表。

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH INERROR TABLES in SCHEMA
 sample_schema;
```

ALTER DATASHARE

變更資料共用的定義。您可以使用 ALTER DATASHARE 來新增物件或移除物件。您只能變更目前資料庫中的資料共用。在資料共用中新增或移除相關聯資料庫中的物件。在要新增或移除的資料共用物件上有所需許可的資料共用擁有者可以修改資料共用。

所需權限

以下是 ALTER DATASHARE 所需的權限：

- 超級使用者。
- 具有 ALTER DATASHARE 權限的使用者。
- 在資料共用上具有 ALTER 或 ALL 權限的使用者。
- 若要將特定物件新增至資料共用，使用者必須要有物件的權限。在這種情況下，使用者必須是物件的擁有者，或具有物件的 SELECT、USAGE 或 ALL 權限。

語法

下列語法說明如何在資料共用中新增或移除物件。

```
ALTER DATASHARE datashare_name { ADD | REMOVE } {
TABLE schema.table [, ...]
| SCHEMA schema [, ...]
| FUNCTION schema.sql_udf (argtype,...) [, ...]
| ALL TABLES IN SCHEMA schema [, ...]
```

```
| ALL FUNCTIONS IN SCHEMA schema [, ...] }
```

下列語法說明如何設定資料共用的屬性。

```
ALTER DATASHARE datashare_name {
[ SET PUBLICACCESSIBLE [=] TRUE | FALSE ]
[ SET INCLUDENEW [=] TRUE | FALSE FOR SCHEMA schema ] }
```

參數

datashare_name

要修改的資料共用名稱。

ADD | REMOVE

指定是否要在資料共用中新增或移除物件的子句。

TABLE schema.table [, ...]

指定結構描述中要新增至資料共用的資料表或檢視名稱。

SCHEMA schema [, ...]

要新增至資料共用的結構描述名稱。

FUNCTION schema.sql_udf (argtype,...) [, ...]

要新增至資料共用且具有引數類型的使用者定義 SQL 函數名稱。

ALL TABLES IN SCHEMA schema [, ...]

指定是否將指定結構描述中的所有資料表和檢視新增至資料共用的子句。

ALL FUNCTIONS IN SCHEMA schema [, ...] }

指定將指定結構描述中所有函數新增至資料共用的子句。

[SET PUBLICACCESSIBLE [=] TRUE | FALSE]

該子句會指定資料共用是否可以共用至可公開存取的叢集。

[SET INCLUDENEW [=] TRUE | FALSE FOR SCHEMA schema]

該子句會指定是否要將指定結構描述中建立的任何未來資料表、檢視或 SQL 使用者定義函數 (UDF) 新增至資料共用。目前指定結構描述中的資料表、檢視或 SQL UDF 不會新增至資料共用。

只有超級用戶可以為每個「資料共用-結構描述」配對修改此屬性。INCLUDENEW 子句會預設為 false。

ALTER DATASHARE 使用須知

- 下列使用者可以修改資料共用：
  - 超級使用者
  - 資料共用的擁有者
  - 在資料共用上具有 ALTER 或 ALL 權限的使用者。
- 若要將特定物件新增至資料共用，使用者必須要有物件的正確權限。使用者必須是物件的擁有者，或具有物件的 SELECT、USAGE 或 ALL 權限。
- 您可以共用結構描述、資料表、一般檢視、近期繫結檢視、具體化視觀表，以及 SQL 使用者定義函數 (UDF)。在結構描述中新增物件之前，請先將結構描述新增至資料共用。

  當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。您必須明確地新增這些物件。
- 我們建議您在開啟可公開存取設定的情況下建立 AWS Data Exchange 資料存取。
- 一般而言，我們建議您不要使用 ALTER AWS Data Exchange DATASHARE 陳述式來變更資料識別，以關閉公用存取功能。如果 AWS 帳戶 這樣做，如果其叢集可公開存取，則可以存取資料清單的存取權限將失去存取權。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。有關此建議的例外情況，請參閱以下內容。

  下列範例顯示在關閉設定的情況下建立 AWS Data Exchange 資料清單時發生錯誤。

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
ERROR:  Alter of ADX-managed datashare salesshare requires session variable
 datashare_break_glass_session_var to be set to value 'c670ba4db22f4b'
```

  若要允許變更 AWS Data Exchange 資料清單關閉可公開存取的設定，請設定下列變數，然後再次執行 ALTER DATASHARE 陳述式。

```
SET datashare_break_glass_session_var to 'c670ba4db22f4b';
```

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
```

  在這種情況下，Amazon Redshift 會產生隨機的一次性值來設定工作階段變數，以允許對 AWS Data Exchange 資料共用執行 ALTER DATASHARE SET PUBLICACCESSIBLE FALSE。

範例

下列範例會將public結構描述新增至資料清單。salesshare

```
ALTER DATASHARE salesshare ADD SCHEMA public;
```

下列範例會將 public.tickit_sales_redshift 資料表新增至 salesshare 資料共用。

```
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
```

下列範例會將所有資料表新增至 salesshare 資料共用。

```
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

下列範例會將 public.tickit_sales_redshift 資料表從 salesshare 資料共用中移除。

```
ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
```

ALTER DEFAULT PRIVILEGES

定義要套用至指定使用者 future 建立之物件的預設存取權限集。根據預設，使用者只能變更自己擁有的預設存取許可。只有超級使用者可以為其他使用者指定預設權限。

您可將預設權限套用至角色、使用者或使用者群組。您可以為目前資料庫中建立的所有物件或僅在指定結構描述中建立的物件設定全域預設權限。

預設權限僅適用於新物件。執行 ALTER 預設權限不會變更現有物件的權限。若要授與資料庫或結構描述中任何使用者建立的所有目前和 future 物件的權限，請參閱 Scoped 權限。

若要檢視有關資料庫使用者預設權限的資訊，請查詢 PG_DEFAULT_ACL 系統目錄資料表。

如需權限的相關資訊，請參閱 GRANT。

所需權限

以下是 ALTER DEFAULT PRIVILEGES 所需的權限：

- 超級使用者

- 具有 ALTER DEFAULT PRIVILEGES 權限的使用者
- 變更自身預設存取權限的使用者
- 為其有權存取的結構描述設定權限的使用者

語法

```
ALTER DEFAULT PRIVILEGES
    [ FOR USER target_user [, ...] ]
    [ IN SCHEMA schema_name [, ...] ]
    grant_or_revoke_clause

where grant_or_revoke_clause is one of:

GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | TRUNCATE } [,...] |
 ALL [ PRIVILEGES ] }
 ON TABLES
 TO { user_name [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
 ON FUNCTIONS
 TO { user_name [ WITH GRANT OPTION ] |  ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
 ON PROCEDURES
 TO { user_name [ WITH GRANT OPTION ] |  ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

REVOKE [ GRANT OPTION FOR ] { { SELECT | INSERT | UPDATE | DELETE | REFERENCES |
 TRUNCATE } [,...] | ALL [ PRIVILEGES ] }
 ON TABLES
 FROM user_name [, ...] [ RESTRICT ]

REVOKE  { { SELECT | INSERT | UPDATE | DELETE | REFERENCES | TRUNCATE } [,...] | ALL
 [ PRIVILEGES ] }
 ON TABLES
 FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ] { EXECUTE | ALL [ PRIVILEGES ] }
 ON FUNCTIONS
 FROM user_name [, ...] [ RESTRICT ]
```

```
REVOKE { EXECUTE | ALL [ PRIVILEGES ] }
 ON FUNCTIONS
 FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ] { EXECUTE | ALL [ PRIVILEGES ] }
 ON PROCEDURES
 FROM user_name [, ...] [ RESTRICT ]

REVOKE { EXECUTE | ALL [ PRIVILEGES ] }
 ON PROCEDURES
 FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]
```

參數

FOR USER target_user

選用。為其定義預設權限的使用者名稱。只有超級使用者能夠指定其他使用者的預設權限。預設值是目前使用者。

IN SCHEMA schema_name

選用。如果出現 IN SCHEMA 子句，指定的預設權限就會套用至所指定 schema_name 中建立的新物件。在此情況下，做為 ALTER DEFAULT PRIVILEGES 目標的使用者或使用者群組必須具備所指定結構描述的 CREATE 權限。結構描述專屬的預設權限會新增至現有的全域預設權限。根據預設，預設權限會全域套用至整個資料庫。

GRANT

授與指定使用者所建立之所有新表格和檢視表、函數或預存程序之指定使用者或群組的權限集。您可以使用 GRANT 子句設定與使用 GRANT 命令時相同的權限和選項。

WITH GRANT OPTION

指出接收權限的使用者可以接著將相同的權限授予他人的子句。您無法將 WITH GRANT OPTION 授予群組或 PUBLIC。

TO user_name | ROLE role_name | GROUP group_name

使用者、角色或使用者群組的名稱，將對其套用指定的預設權限。

REVOKE

對於指定使用者所建立的所有新資料表、函數或預存程序，要撤銷指定使用者或群組的權限集。您可以使用 REVOKE 子句設定與使用 REVOKE 命令時相同的權限和選項。

GRANT OPTION FOR

這個子句只會撤銷對其他使用者授予指定權限的選項，而不會撤銷權限本身。您無法撤銷群組或 PUBLIC 的 GRANT OPTION。

FROM user_name | ROLE role_name | GROUP group_name

使用者、角色或使用者群組的名稱，預設將撤銷其指定權限。

RESTRICT

RESTRICT 選項只會撤銷使用者直接授予的權限。此為預設值。

範例

假設您要允許使用者群組中的任何使用者檢視report_readers使用者建立的所有資料表和檢視表report_admin。在此情況下，請以超級使用者身分執行下列命令。

```
alter default privileges for user report_admin grant select on tables to group
 report_readers;
```

在下列範例中，第一個命令會授與您建立的所有新資料表和檢視的 SELECT 權限。

```
alter default privileges grant select on tables to public;
```

下列範例會對 sales_admin 使用者群組授予您在 sales 結構描述中建立的所有新資料表和檢視的 INSERT 權限。

```
alter default privileges in schema sales grant insert on tables to group sales_admin;
```

以下範例會反轉前述範例中的 ALTER DEFAULT PRIVILEGES 命令。

```
alter default privileges in schema sales revoke insert on tables from group
 sales_admin;
```

根據預設，PUBLIC 使用者群組具有所有新的使用者定義功能的執行許可。若要撤銷新功能的 public 執行許可，然後僅將執行許可授予 dev_test 使用者群組，請執行下列命令。

```
alter default privileges revoke execute on functions from public;
alter default privileges grant execute on functions to group dev_test;
```

ALTER EXTERNAL VIEW (預覽)

這是適用於 Amazon Redshift 的資料目錄中的發行前版本文件檢視，屬於預覽版本。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。 AWS 區域 會列出目前帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的建立叢集。
5. 選擇建立叢集按鈕以建立預覽叢集。

   > **Note**
   >
   > `preview_2023` 軌跡是最近可用的預覽軌跡。此軌跡僅支援使用 RA3 節點類型建立叢集。不支援節點類型 DC2 和任何較舊的節點類型。

6. 當您的預覽叢集可用時，請使用 SQL 用戶端載入和查詢資料。

Data Catalog 視觀表預覽功能僅適用於以下區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)
- 亞太區域 (東京) (ap-northeast-1)

- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

您也可以建立預覽工作群組來測試 Data Catalog 視觀表。您無法在生產環境中使用這些功能，也無法將工作群組移至另一個工作群組。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。如需如何建立預覽工作群組的指示，請參閱建立預覽工作群組。

使用 ALTER EXTERNAL VIEW 命令更新外部檢視。視您使用的參數而定，也可以參考此檢視的其他 SQL 引擎 (例如 Amazon Athena 和 Amazon EMR Spark) 可能會受到影響。如需有關 Data Catalog 視觀表的詳細資訊，請參閱建立 Data Catalog 視觀表 (預覽)。

語法

```
ALTER EXTERNAL VIEW schema_name.view_name
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
 external_schema_name.view_name}
[FORCE] { AS (query_definition) | REMOVE DEFINITION }
```

參數

schema_name.view_name

附加到 AWS Glue 資料庫的結構描述，後面接著檢視的名稱。

catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name | external_schema_name.view_name

變更視觀表時要使用的結構描述標記法。您可以指定使用 AWS Glue Data Catalog您建立的 Glue 資料庫或您建立的外部結構描述。如需詳細資訊，請參閱CREATE DATABASE 和 CREATE EXTERNAL SCHEMA。

FORCE

即使資料表中參照的物件與其他 SQL 引擎不一致，是否 AWS Lake Formation 應更新檢視的定義。如果 Lake Formation 更新檢視，則對於其他 SQL 引擎來說，該檢視會被視為過時，直到這些引擎也更新為止。

AS query_definition

Amazon Redshift 執行以變更檢視的 SQL 查詢的定義。

REMOVE DEFINITION

是否捨棄並重新建立檢視。必須捨棄檢視並重新建立，才能將其標記為 PROTECTED。

範例

下列範例會變更名為 sample_schema.glue_data_catalog_view 的 Data Catalog 視觀表。

```
ALTER EXTERNAL VIEW sample_schema.glue_data_catalog_view
FORCE
REMOVE DEFINITION
```

ALTER FUNCTION

重新命名函數或變更擁有者。無論是函數名稱和數據類型是必需的。只有擁有者或超級使用者可以重新命名函數。只有超級用戶可以更改函數的所有者。

語法

```
ALTER FUNCTION function_name ( { [ py_arg_name py_arg_data_type | sql_arg_data_type }
 [ , ... ] ] )
    RENAME TO new_name
```

```
ALTER FUNCTION function_name ( { [ py_arg_name py_arg_data_type | sql_arg_data_type }
 [ , ... ] ] )
    OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
```

參數

function_name

要更改的函數的名稱。在目前搜尋路徑中指定函數的名稱，或使用格式schema_name.function_name來使用特定結構描述。

型別資料類型 | 正方形資料類型

選用。Python 使用者定義函數的輸入引數名稱和資料類型清單，或 SQL 使用者定義函數的輸入引數資料類型清單。

new_name

使用者定義函數的新名稱。

new_owner | CURRENT_USER | SESSION_USER

使用者定義函數的新擁有者。

範例

下列範例會將函數的名稱從變更first_quarter_revenue為quarterly_revenue。

```
ALTER FUNCTION first_quarter_revenue(bigint, numeric, int)
         RENAME TO quarterly_revenue;
```

下列範例會將quarterly_revenue函數的擁有者變更為etl_user。

```
ALTER FUNCTION quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

ALTER GROUP

變更使用者群組。使用此命令可將使用者新增至群組、從群組中刪除使用者，或重新命名群組。

語法

```
ALTER GROUP group_name
{
ADD USER username [, ... ] |
DROP USER username [, ... ] |
RENAME TO new_name
}
```

參數

group_name

要修改的使用者群組名稱。

ADD

將使用者新增至使用者群組。

DROP

從使用者群組中移除使用者。

username

要新增至群組或從群組中刪除的使用者名稱。

RENAME TO

重新命名使用者群組。開頭為兩個底線的群組名稱保留供 Amazon Redshift 內部使用。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

new_name

使用者群組的新名稱。

範例

以下範例會將名為 DWUSER 的使用者新增至 ADMIN_GROUP 群組。

```
ALTER GROUP admin_group
ADD USER dwuser;
```

以下範例會將群組 ADMIN_GROUP 重新命名為 ADMINISTRATORS。

```
ALTER GROUP admin_group
RENAME TO administrators;
```

以下範例會將兩個使用者新增至 ADMIN_GROUP 群組。

```
ALTER GROUP admin_group
ADD USER u1, u2;
```

以下範例會將兩個使用者從 ADMIN_GROUP 群組中捨棄。

```
ALTER GROUP admin_group
DROP USER u1, u2;
```

ALTER IDENTITY PROVIDER

修改身分提供者以指派新參數和值。當您執行此命令時，所有先前設定的參數值都會先刪除，然後再指派新值。只有超級使用者可以修改身分提供者。

語法

```
ALTER IDENTITY PROVIDER identity_provider_name
[PARAMETERS parameter_string]
[NAMESPACE namespace]
[IAM_ROLE iam_role]
[DISABLE | ENABLE]
```

參數

identity_provider_name

新身分提供者的名稱。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

parameter_string

包含正確格式化 JSON 物件的字串，其中包含特定身分提供者所需的參數和值。

命名空間

組織命名空間。

阿姆角色

為 IAM 身分中心連線提供許可的 IAM 角色。此參數僅適用於身分識別提供者類型為時。 AWSIDC

停用或啟用

開啟或關閉身分識別提供者。預設值為「啟用」

範例

下列範例會修改名為 oauth_standard 的身分提供者。它特別適用於當 Microsoft Azure AD 是身分識別提供者。

```
ALTER IDENTITY PROVIDER oauth_standard
PARAMETERS '{"issuer":"https://sts.windows.net/2sdfdsf-d475-420d-b5ac-667adad7c702/",
"client_id":"87f4aa26-78b7-410e-bf29-57b39929ef9a",
"client_secret":"BUAH~ewrqewrqwerUUY^%tHe1oNZShoiU7",
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"]
}'
```

下列範例顯示如何設定身分識別提供者命名空間。這可能適用於 Microsoft Azure AD，如果它遵循上一個範例的陳述式，或是另一個身分識別提供者。如果您已透過受管應用程式設定連線，也可以套用至將現有 Amazon Redshift 佈建叢集或 Amazon Redshift 無伺服器工作群組連線至 IAM 身分中心的案例。

```
ALTER IDENTITY PROVIDER "my-redshift-idc-application"
NAMESPACE 'MYCO';
```

下列範例會設定 IAM 角色，並適用於設定 Redshift 與 IAM 身分中心整合的使用案例。

```
ALTER IDENTITY PROVIDER "my-redshift-idc-application"
IAM_ROLE 'arn:aws:iam::123456789012:role/myadministratorrole';
```

如需有關從 Redshift 設定 IAM 身分中心連線的詳細資訊，請參閱 Connect Redshift 與 IAM 身分中心連線，為使用者提供單一登入體驗。

停用身分識別提供者

下列範例陳述式顯示如何停用身分識別提供者。停用時，身分識別提供者的聯合身分使用者必須重新啟用叢集才能登入叢集。

```
ALTER IDENTITY PROVIDER "redshift-idc-app" DISABLE;
```

ALTER MASKING POLICY

修改現有的動態資料遮罩政策。如需動態資料遮罩的相關資訊，請參閱 動態資料遮罩。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以修改遮罩政策。

語法

```
ALTER MASKING POLICY policy_name
   USING (masking_expression);
```

參數

policy_name

遮罩政策的名稱。這必須是資料庫中已存在的遮罩政策名稱。

masking_expression

用來轉換目標資料欄的 SQL 運算式。其可以使用資料操作函數 (例如字符串操作函數) 編寫，或與使用 SQL、Python 或 AWS Lambda編寫的使用者定義函數一起編寫。

運算式必須與原始運算式的輸入資料欄和資料類型相符。例如，如果原始遮罩政策的輸入資料欄是 `sample_1 FLOAT` 和 `sample_2 VARCHAR(10)`，您將無法修改遮罩政策以取得第三個資料欄，或讓政策採用 FLOAT 和 BOOLEAN。如果您使用常數做為遮罩運算式，則必須將其明確轉換為符合輸入類型的類型。

對於在遮罩運算式中使用的任何使用者定義函數，您都必須擁有 USAGE 權限。

ALTER MATERIALIZED VIEW

啟用具體化視觀表的自動重新整理。

語法

```
ALTER MATERIALIZED VIEW mv_name
[ AUTO REFRESH { YES | NO } ]
[ ROW LEVEL SECURITY { ON | OFF } [ CONJUNCTION TYPE { AND | OR } ] [FOR DATASHARES] ];
```

參數

mv_name

具體化視觀表的名稱。

AUTO REFRESH { YES | NO }

開啟或關閉具體化視觀表自動重新整理的子句。如需具體化視觀表自動重新整理的相關資訊，請參閱 重新整理具體化視觀表。

ROW LEVEL SECURITY { ON | OFF } [CONJUNCTION TYPE { AND | OR }] [FOR DATASHARES]

開啟或關閉關係的資料列層級安全性的子句。

開啟關係的資料列層級安全性時，您只能讀取資料列層級安全政策允許您存取的資料列。如果沒有任何政策授予您關係存取權，您就無法看到關聯中的任何資料列。只有超級使用者和具有 `sys:secadmin` 角色的使用者或角色可以設定 ROW LEVEL SECURITY 子句。如需詳細資訊，請參閱 資料列層級安全性。

- [CONJUNCTION TYPE { AND | OR }]

  一個子句，讓您為關係選擇資料列層級安全政策的結合類型。將多個資料列層級安全政策附加至關係時，您可以將政策與 AND 或 OR 子句結合使用。根據預設，Amazon Redshift 將 RLS 政策與 AND 子句結合在一起。具有 sys:secadmin 角色的超級使用者、使用者或角色可以使用此子句來定義關係之資料列層級安全政策的組合類型。如需詳細資訊，請參閱 每個使用者結合多個政策。

- FOR DATASHARES

  該子句會決定是否可透過資料共用存取受 RLS 保護的關係。預設的情況是無法透過資料共用存取受 RLS 保護的關係。搭配此子句執行的 ALTER MATERIALIZED VIEW ROW LEVEL SECURITY 命令只會影響關係的資料共用存取屬性。ROW LEVEL SECURITY 屬性並未變更。

  如果您讓受 RLS 保護的關係可以透過資料庫存取，則該關係在取用者端資料庫中不會有資料列層級的安全性。關係會在生產者端保留其 RLS 屬性。

範例

下列範例可讓 tickets_mv 具體化視觀表自動重新整理。

```
ALTER MATERIALIZED VIEW tickets_mv AUTO REFRESH YES
```

編輯器樣式和排序碼範例

本主題中的範例說明如何使用 ALTER 具體化視觀表來執行 DISTSTYLE 和 SORTKEY 變更。

下列範例查詢會示範如何使用範例基底資料表來變更 DISTSTYLE 金鑰 DISTKEY 資料行：

```
CREATE TABLE base_inventory(
inv_date_sk int4 not null,
inv_item_sk int4 not null,
inv_warehouse_sk int4 not null,
inv_quantity_on_hand int4
);

INSERT INTO base_inventory VALUES(1,1,1,1);

CREATE MATERIALIZED VIEW inventory DISTSTYLE EVEN
as SELECT * FROM base_inventory;
SELECT "table", DISTSTYLE FROM svv_table_info WHERE "table" = 'inventory';
```

```
ALTER MATERIALIZED VIEW inventory ALTER DISTSTYLE KEY DISTKEY inv_warehouse_sk;
SELECT "table", DISTSTYLE FROM svv_table_info where "table" = 'inventory';

ALTER MATERIALIZED VIEW inventory ALTER DISTKEY inv_item_sk;
SELECT "table", diststyle from svv_table_info where "table" = 'inventory';
```

將具體化視觀表更改為 DISTSTYLE 全部：

```
CREATE TABLE base_inventory(
inv_date_sk int4 not null,
inv_item_sk int4 not null,
inv_warehouse_sk int4 not null,
inv_quantity_on_hand int4
);

INSERT INTO base_inventory values(1,1,1,1);

CREATE MATERIALIZED VIEW inventory DISTSTYLE EVEN
as SELECT * FROM base_inventory;

SELECT "table", DISTSTYLE FROM svv_table_info WHERE "table" = 'inventory';
```

下列命令會使用範例基底資料表來顯示 ALTER 具體化視觀表 SORTKEY 範例：

```
CREATE MATERIALIZED VIEW base_inventory (c0 int, c1 int);

CREATE MATERIALIZED VIEW inventory
interleaved sortkey(c0, c1)
as SELECT * FROM base_inventory;

SELECT "table", sortkey1 FROM svv_table_info WHERE "table" = 'inventory';

ALTER MATERIALIZED VIEW t1 alter sortkey(c0, c1);
SELECT "table", diststyle, sortkey_num FROM svv_table_info WHERE "table" = 'inventory';

ALTER MATERIALIZED VIEW t1 alter sortkey none;
SELECT "table", diststyle, sortkey_num FROM svv_table_info WHERE "table" = 'inventory';

ALTER MATERIALIZED VIEW t1 alter sortkey(c0);
SELECT "table", diststyle, sortkey_num FROM svv_table_info WHERE "table" = 'inventory';
```

ALTER RLS POLICY

修改資料表上現有的資料列層級安全性政策。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以修改政策。

語法

```
ALTER RLS POLICY policy_name
USING ( using_predicate_exp );
```

參數

policy_name

政策的名稱。

USING (using_predicate_exp)

指定套用至查詢中 WHERE 子句的篩選條件。Amazon Redshift 會在查詢層級使用者述詞之前套用政策述詞。例如，**current_user = 'joe' and price > 10** 會限制 Joe 只能查看價格大於 $10 的記錄。

在用來以 policy_name 名稱建立政策的 CREATE RLS POLICY 陳述式中，運算式可以存取其中 WITH 子句中宣告的變數。

範例

下列範例會修改 RLS 政策。

```
-- First create an RLS policy that limits access to rows where catgroup is 'concerts'.
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'concerts');

-- Then, alter the RLS policy to only show rows where catgroup is 'piano concerts'.
ALTER RLS POLICY policy_concerts
USING (catgroup = 'piano concerts');
```

ALTER ROLE

重新命名角色或變更擁有者。如需 Amazon Redshift 系統定義角色的清單，請參閱 the section called "Amazon Redshift 系統定義角色"。

所需的許可

以下是 ALTER ROLE 所需的權限：

- 超級使用者
- 具有 ALTER ROLE 許可的使用者

語法

```
ALTER ROLE role [ WITH ]
  { { RENAME TO role } | { OWNER TO user_name } }[, ...]
  [ EXTERNALID TO external_id ]
```

參數

角色

要修改的角色名稱。

RENAME TO

角色的新名稱。

OWNER TO user_name

角色的新擁有者。

EXTERNALID TO external_id

角色的新外部 ID，與身分提供者相關聯。如需詳細資訊，請參閱 Amazon Redshift 的原生身分提供者 (IdP) 聯合合。

範例

下列範例會將角色的名稱從 `sample_role1` 變更為 `sample_role2`。

```
ALTER ROLE sample_role1 WITH RENAME TO sample_role2;
```

下列範例會變更角色的擁有者。

```
ALTER ROLE sample_role1 WITH OWNER TO user1
```

ALTER ROLE 的語法類似於下面的 ALTER PROCEDURE。

```
ALTER PROCEDURE first_quarter_revenue(bigint, numeric) RENAME TO quarterly_revenue;
```

下列範例會將程序的擁有者變更為 etl_user。

```
ALTER PROCEDURE quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

下列範例會使用與身分提供者相關聯的新外部 ID 來更新角色 sample_role1。

```
ALTER ROLE sample_role1 EXTERNALID TO "XYZ456";
```

ALTER PROCEDURE

重新命名程序或變更擁有者。程序名稱和資料類型 (或簽章) 都是必要的。只有擁有者或超級使用者才能重新命名程序。只有超級使用者才能變更程序的擁有者。

語法

```
ALTER PROCEDURE sp_name [ ( [ [ argname ] [ argmode ] argtype [, ...] ] ) ]
    RENAME TO new_name
```

```
ALTER PROCEDURE sp_name [ ( [ [ argname ] [ argmode ] argtype [, ...] ] ) ]
    OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
```

參數

sp_name

要修改的程序的名稱。在目前的搜尋路徑中僅指定程序的名稱，或使用格式 schema_name.sp_procedure_name 來使用專屬結構描述。

[argname] [argmode] argtype

引數名稱、引數模式和資料類型的清單。只有用於識別預存程序的輸入資料類型是必要的。或者，您可以提供用於建立程序的完整簽章，包括輸入和輸出參數及其模式。

new_name

預存程序的新名稱。

new_owner | CURRENT_USER | SESSION_USER

預存程序的新擁有者。

範例

下列範例會將程序的名稱從 first_quarter_revenue 變更為 quarterly_revenue。

```
ALTER PROCEDURE first_quarter_revenue(volume INOUT bigint, at_price IN numeric,
 result OUT int) RENAME TO quarterly_revenue;
```

此範例相當於下列命令。

```
ALTER PROCEDURE first_quarter_revenue(bigint, numeric) RENAME TO quarterly_revenue;
```

下列範例會將程序的擁有者變更為 etl_user。

```
ALTER PROCEDURE quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

ALTER SCHEMA

變更現有結構描述的定義。使用此命令可重新命名結構描述或變更結構描述的擁有者。例如，重新命名現有的結構描述，以便在您打算建立該結構描述的新版本時，保留該結構描述的備份複本。如需結構描述的相關資訊，請參閱 CREATE SCHEMA。

若要檢視設定的結構描述配額，請參閱 SVV_SCHEMA_QUOTA_STATE。

若要檢視超出結構描述配額的記錄，請參閱 STL_SCHEMA_QUOTA_VIOLATIONS。

所需權限

以下是 ALTER SCHEMA 所需的權限：

- 超級使用者
- 具有 ALTER SCHEMA 權限的使用者
- 結構描述擁有者

當您變更結構描述名稱時，請務必更新使用舊名稱的物件 (例如預存程序或具體化視觀表)，才能使用新名稱。

語法

```
ALTER SCHEMA schema_name
{
RENAME TO new_name |
OWNER TO new_owner |
QUOTA { quota [MB | GB | TB] | UNLIMITED }
}
```

參數

schema_name

要修改的資料庫結構描述名稱。

RENAME TO

重新命名結構描述的子句。

new_name

結構描述的新名稱。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

OWNER TO

變更結構描述擁有者的子句。

new_owner

結構描述的新擁有者。

QUOTA

所指定之結構描述可以使用的磁碟空間數上限。此空間是所指定之結構描述下所有資料表的集體大小。Amazon Redshift 會將選取的值轉換為 MB。GB 是您未指定值時的預設測量單位。

如需設定結構描述配額的相關資訊，請參閱 CREATE SCHEMA。

範例

下列範例會將 SALES 結構描述重新命名為 US_SALES。

```
alter schema sales
rename to us_sales;
```

下列範例會將 US_SALES 結構描述的擁有權提供給使用者 DWUSER。

```
alter schema us_sales
owner to dwuser;
```

下列範例會將配額變更為 300 GB，並移除配額。

```
alter schema us_sales QUOTA 300 GB;
alter schema us_sales QUOTA UNLIMITED;
```

ALTER SYSTEM

變更 Amazon Redshift 叢集或 Redshift Serverless 工作群組的系統層級組態選項。

所需權限

下列其中一種使用者類型可以執行 ALTER SYSTEM 命令：

- 超級使用者
- 管理員使用者

語法

```
ALTER SYSTEM SET system-level-configuration = {true| t | on | false | f | off}
```

參數

system-level-configuration

系統層級組態 有效值：`data_catalog_auto_mount` 和 `metadata_security`。

{true| t | on | false | f | off}

用於啟用或停用系統層級組態的值。true、t 或 on 表示要啟用組態。false、f 或 off 表示要停用組態。

使用須知

對於佈建的叢集，data_catalog_auto_mount 的變更會在下次重新啟動叢集時生效。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的重新啟動叢集。

對於無伺服器工作群組，data_catalog_auto_mount 的變更不會立即生效。

範例

下列範例會開啟自動掛載 AWS Glue Data Catalog。

```
ALTER SYSTEM SET data_catalog_auto_mount = true;
```

下列範例會開啟中繼資料安全性。

```
ALTER SYSTEM SET metadata_security = true;
```

設定預設的識別名稱空間

此範例特定於使用身分識別提供者。您可以將 Redshift 與 IAM 身分中心和身分識別提供者整合，以集中 Redshift 和其他服務的身分識別管理。 AWS

下列範例顯示如何設定系統的預設識別名稱空間。接下來這麼做會讓執行 GRANT 和 CREATE 陳述式變得更加簡單，因為您不需要將命名空間納入為每個識別的前置詞。

```
ALTER SYSTEM SET default_identity_namespace = 'MYCO';
```

執行命令之後，您可以執行如下所示的陳述式：

```
GRANT SELECT ON TABLE mytable TO alice;

GRANT UPDATE ON TABLE mytable TO salesrole;

CREATE USER bob password 'md50c983d1a624280812631c5389e60d48c';
```

設置默認身份命名空間的效果是，每個身份都不需要它作為前綴。在此範例中，alice會取代為MYCO:alice。這發生在包含任何身份的情況下。如需有關透過 Redshift 使用身分識別提供者的詳細資訊，請參閱將 Redshift 與 IAM 身分中心連線，為使用者提供單一登入體驗。

如需使用 IAM 身分中心與 Redshift 組態相關之設定的詳細資訊，請參閱SET和。ALTER IDENTITY PROVIDER

ALTER TABLE

此命令會變更 Amazon Redshift 資料表或 Amazon Redshift Spectrum 外部資料表的定義。此命令會更新 CREATE TABLE 或 CREATE EXTERNAL TABLE 設定的值和屬性。

您不能在交易區塊 (BEGIN ... END) 內的外部資料表上執行 ALTER TABLE。如需交易的相關資訊，請參閱 可序列化隔離。

ALTER TABLE 會鎖定用於讀取和寫入操作的資料表，直到封閉 ALTER TABLE 操作的交易完成為止，除非在文件中明確說明您可以在修改資料表時，查詢資料或對資料表執行其他操作。

所需權限

修改資料表的使用者需要適當的權限，才能成功執行命令。根據 ALTER TABLE 命令的不同，使用者可能需要下列其中一項權限。

- 超級使用者
- 具有 ALTER TABLE 權限的使用者
- 具有結構描述的 USAGE 權限的資料表擁有者

語法

```
ALTER TABLE table_name
{
ADD table_constraint
| DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
| OWNER TO new_owner
| RENAME TO new_name
| RENAME COLUMN column_name TO new_name
| ALTER COLUMN column_name TYPE updated_varchar_data_type_size
| ALTER COLUMN column_name ENCODE new_encode_type
| ALTER COLUMN column_name ENCODE encode_type,
| ALTER COLUMN column_name ENCODE encode_type, .....;
```

```
| ALTER DISTKEY column_name
| ALTER DISTSTYLE ALL
| ALTER DISTSTYLE EVEN
| ALTER DISTSTYLE KEY DISTKEY column_name
| ALTER DISTSTYLE AUTO
| ALTER [COMPOUND] SORTKEY ( column_name [,...] )
| ALTER SORTKEY AUTO
| ALTER SORTKEY NONE
| ALTER ENCODE AUTO
| ADD [ COLUMN ] column_name column_type
  [ DEFAULT default_expr ]
  [ ENCODE encoding ]
  [ NOT NULL | NULL ]
  [ COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE } ] |
| DROP [ COLUMN ] column_name [ RESTRICT | CASCADE ]
| ROW LEVEL SECURITY { ON | OFF } [ CONJUNCTION TYPE { AND | OR } ] [ FOR DATASHARES ]}

where table_constraint is:

[ CONSTRAINT constraint_name ]
{ UNIQUE ( column_name [, ... ] )
| PRIMARY KEY ( column_name [, ... ] )
| FOREIGN KEY (column_name [, ... ] )
   REFERENCES  reftable [ ( refcolumn ) ]}

The following options apply only to external tables:

SET LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file' }
| SET FILE FORMAT format |
| SET TABLE PROPERTIES ('property_name'='property_value')
| PARTITION ( partition_column=partition_value [, ...] )
  SET LOCATION { 's3://bucket/folder' |'s3://bucket/manifest_file' }
| ADD [IF NOT EXISTS]
    PARTITION ( partition_column=partition_value [, ...] ) LOCATION
 { 's3://bucket/folder' |'s3://bucket/manifest_file' }
    [, ... ]
| DROP PARTITION ( partition_column=partition_value [, ...] )
```

若要減少執行 ALTER TABLE 命令的時間，你可以結合 ALTER TABLE 命令的部分子句。

Amazon Redshift 支援 ALTER TABLE 子句的以下組合：

```
ALTER TABLE tablename ALTER SORTKEY (column_list), ALTER DISTKEY column_Id;
ALTER TABLE tablename ALTER DISTKEY column_Id, ALTER SORTKEY (column_list);
```

```
ALTER TABLE tablename ALTER SORTKEY (column_list), ALTER DISTSTYLE ALL;
ALTER TABLE tablename ALTER DISTSTYLE ALL, ALTER SORTKEY (column_list);
```

參數

table_name

要修改的資料表名稱。僅指定資料表的名稱，或使用格式 schema_name.table_name 來使用專屬結構描述。外部資料表必須以外部結構描述名稱限定。如果您要使用 ALTER TABLE 陳述式重新命名檢視或變更其擁有者，則也可以指定檢視名稱。資料表名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 多位元組字元，最長可達 4 個位元組。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

ADD table_constraint

新增指定限制條件至資料表的子句。如需有效 table_constraint 值得說明，請參閱 CREATE TABLE。

> **Note**
>
> 您無法將主索引鍵限制條件新增至可為 null 的資料欄。如果資料欄原本是以 NOT NULL 限制條件建立，則您可以新增主索引鍵限制條件。

DROP CONSTRAINT constraint_name

從資料表中刪除具名限制條件的子句。若要刪除限制條件，請指定限制條件名稱，而非限制條件類型。若要檢視資料表限制條件名稱，請執行下列查詢。

```
select constraint_name, constraint_type
from information_schema.table_constraints;
```

RESTRICT

僅移除指定限制條件的子句。RESTRICT 是 DROP CONSTRAINT 的選項。RESTRICT 不能搭配 CASCADE 使用。

CASCADE

此子句會移除指定限制條件及相依於該限制條件的任何項目。CASCADE 是 DROP CONSTRAINT 的選項。CASCADE 不能搭配 RESTRICT 使用。

OWNER TO new_owner

此子句會將資料表 (或檢視) 的擁有者變更為 new_owner 值。

RENAME TO new_name

此子句會將資料表 (或檢視) 重新命名為 new_name 中指定的值。資料表名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。

您無法將永久資料表重新命名為開頭為 '#' 的名稱。開頭為 '#' 的資料表名稱代表暫時資料表。

您無法重新命名外部資料表。

ALTER COLUMN column_name TYPE updated_varchar_data_type_size

子句，會變更定義為 VARCHAR 資料類型的資料欄大小。此子句僅支援修改 VARCHAR 資料類型的大小。考量下列限制：

- 您無法修改具有以下壓縮編碼的欄位：BYTEDICT、RUNLENGTH、TEXT255 或 TEXT32K。
- 您無法將大小降低至小於現有資料的大小上限。
- 您無法修改含預設值的資料欄。
- 您無法修改具有 UNIQUE、PRIMARY KEY 或 FOREIGN KEY 的資料欄。
- 您無法在交易區塊 (BEGIN ... END) 內改變資料欄。如需交易的相關資訊，請參閱 可序列化隔離。

ALTER COLUMN column_name ENCODE new_encode_type

變更資料欄壓縮編碼的子句。如果您指定資料欄的壓縮編碼，資料表就不會再設定為 ENCODE AUTO。如需壓縮編碼的詳細資訊，請參閱 使用資料欄壓縮。

當您變更資料欄的壓縮編碼時，資料表仍可供查詢。

考量下列限制：

- 您無法將資料欄的編碼修改為目前資料行定義的編碼。
- 您無法在具有交錯排序索引鍵的資料表中修改資料行的編碼。

ALTER COLUMN column_name ENCODE encode_type, ALTER COLUMN column_name ENCODE encode_type, ...;

該子句可變更單一命令中多個資料欄的壓縮編碼。如需壓縮編碼的詳細資訊，請參閱 使用資料欄壓縮。

當您變更資料欄的壓縮編碼時，資料表仍可供查詢。

考量下列限制：

- 您無法在單一命令中多次將資料欄變更為相同或不同的編碼類型。
- 您無法將資料欄的編碼修改為目前資料行定義的編碼。
- 您無法在具有交錯排序索引鍵的資料表中修改資料行的編碼。

ALTER DISTSTYLE ALL

將資料表的現有分佈樣式變更為 ALL 的子句。考慮下列各項：

- 無法同時在相同資料表上執行 ALTER DISTSTYLE、ALTER SORTKEY 和 VACUUM。
  - 如果 VACUUM 目前執行中，然後執行 ALTER DISTSTYLE ALL 會傳回錯誤。
  - 如果 ALTER DISTSTYLE ALL 執行中，則不會在資料表上啟動背景清空。
- 包含交錯排序索引鍵的資料表和暫存資料表不支援 ALTER DISTSTYLE ALL 命令。
- 如果分佈樣式先前已定義為 AUTO，則該資料表不再是自動資料表最佳化的候選項目。

如需 DISTSTYLE ALL 的相關資訊，請參閱 CREATE TABLE。

ALTER DISTSTYLE EVEN

將資料表的現有分佈樣式變更為 EVEN 的子句。考慮下列各項：

- 無法同時在相同資料表上執行 ALTER DISTSYTLE、ALTER SORTKEY 和 VACUUM。
  - 如果 VACUUM 目前執行中，然後執行 ALTER DISTSTYLE EVEN 會傳回錯誤。
  - 如果 ALTER DISTSTYLE EVEN 執行中，則不會在資料表上啟動背景清空。
- 包含交錯排序索引鍵的資料表和暫存資料表不支援 ALTER DISTSTYLE EVEN 命令。
- 如果分佈樣式先前已定義為 AUTO，則該資料表不再是自動資料表最佳化的候選項目。

如需 DISTSTYLE EVEN 的相關資訊，請參閱 CREATE TABLE。

ALTER DISTKEY column_name 或 ALTER DISTSTYLE KEY DISTKEY column_name

會變更用作資料表的分佈索引鍵的資料欄的子句。考慮下列各項：

- VACUUM 和 ALTER DISTKEY 無法並行在相同資料表上執行。
  - 如果 VACUUM 已在執行，則 ALTER DISTKEY 會傳回錯誤。
  - 如果 ALTER DISTKEY 執行中，則不會在資料表上啟動背景清空。
  - 如果 ALTER DISTKEY 執行中，則前景清空會傳回錯誤。
- 您一次僅可以在一個資料表上執行一個 ALTER DISTKEY 命令。
- 包含交錯排序索引鍵的資料表不支援 ALTER DISTKEY 命令。

- 如果分佈樣式先前已定義為 AUTO，則該資料表不再是自動資料表最佳化的候選項目。

指定 DISTSTYLE KEY 時，資料是依 DISTKEY 資料欄中的值分佈。如需 DISTSTYLE 的相關資訊，請參閱 CREATE TABLE。

ALTER DISTSTYLE AUTO

將資料表的現有分佈樣式變更為 AUTO 的子句。

將分佈樣式變更為 AUTO 時，資料表的分佈樣式會設定為下列內容：

- 將具有 DISTSTYLE ALL 的小型資料表轉換為 AUTO(ALL)。
- 將具有 DISTSTYLE EVEN 的小型資料表轉換為 AUTO(ALL)。
- 將具有 DISTSTYLE KEY 的小型資料表轉換為 AUTO(ALL)。
- 將具有 DISTSTYLE ALL 的大型資料表轉換為 AUTO(EVEN)。
- 將具有 DISTSTYLE EVEN 的大型資料表轉換為 AUTO(EVEN)。
- 將具有 DISTSTYLE KEY 的大型資料表轉換為 AUTO(KEY) 並保留 DISTKEY 。在這種情況下，Amazon Redshift 不會對資料表進行任何變更。

如果 Amazon Redshift 判斷新的分佈樣式或索引鍵可以改善查詢效能，那麼 Amazon Redshift 可能會在未來變更資料表的分佈樣式或索引鍵。例如，Amazon Redshift 可能會將 DISTSTYLE 為 AUTO(KEY) 的資料表轉換為 AUTO(EVEN)，反之亦然。如需修改分佈索引鍵時行為 (包括資料重新分佈和鎖定) 的相關資訊，請參閱 Amazon Redshift Advisor 建議。

如需 DISTSTYLE AUTO 的相關資訊，請參閱 CREATE TABLE。

若要檢視資料表的分佈樣式，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱 SVV_TABLE_INFO。若要檢視資料表的 Amazon Redshift Advisor 建議，請查詢 SVV_ALTER_TABLE_RECOMMENDATIONS 系統目錄檢視。如需詳細資訊，請參閱 SVV_ALTER_TABLE_RECOMMENDATIONS。若要檢視 Amazon Redshift 所採取的動作，請查詢 SVL_AUTO_WORKER_ACTION 系統目錄檢視。如需詳細資訊，請參閱 SVL_AUTO_WORKER_ACTION。

ALTER [COMPOUND] SORTKEY (column_name [,...])

此子句可變更或新增用於資料表的排序索引鍵。

當您變更排序索引鍵時，新排序索引鍵或原始排序索引鍵中欄的壓縮編碼可能會變更。如果沒有明確定義資料表的編碼，則 Amazon Redshift 會自動指定壓縮編碼，如下所示：

- 定義為排序索引鍵的資料欄會有指派的 RAW 壓縮。
- 定義為 BOOLEAN、REAL 或 DOUBLE PRECISION 資料類型的資料欄會有指派的 RAW 壓縮。

- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP 或 TIMESTAMPTZ 的資料欄會有指派的 AZ64 壓縮。
- 定義為 CHAR 或 VARCHAR 的資料欄會有指派的 LZO 壓縮。

考慮下列各項：

- 每個資料表的排序索引鍵最多可定義 400 個資料欄。
- 您可以將交錯排序索引鍵修改為複合排序索引鍵或無排序索引鍵。但是，您無法將複合排序索引鍵修改為交錯排序索引鍵。
- 如果排序索引鍵先前已定義為 AUTO，則該資料表不再是自動資料表最佳化的候選項目。
- Amazon Redshift 建議對定義為排序索引鍵的資料欄使用 RAW 編碼 (不壓縮)。當您變更資料欄以選擇其做為排序索引鍵時，資料欄的壓縮會變更為 RAW 壓縮 (不壓縮)。這可能會增加資料表所需的儲存空間。資料表大小的增加幅度取決於特定的資料表定義和資料表內容。如需壓縮的相關資訊，請參閱 壓縮編碼。

當資料載入資料表之後，資料就會依據排序索引鍵的排序載入。當您修改排序索引鍵時，Amazon Redshift 會重新排列資料。如需 SORTKEY 的相關資訊，請參閱CREATE TABLE。

ALTER SORTKEY AUTO

將目標資料表的排序索引鍵變更或新增為 AUTO 的子句。

當您將排序索引鍵修改為 AUTO 時，Amazon Redshift 會保留資料表的現有排序索引鍵。

如果 Amazon Redshift 判斷新的排序索引鍵可以改善查詢效能，那麼 Amazon Redshift 可能會在未來變更資料表的排序索引鍵。

如需 SORTKEY AUTO 的相關資訊，請參閱 CREATE TABLE。

若要檢視資料表的排序索引鍵，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱 SVV_TABLE_INFO。若要檢視資料表的 Amazon Redshift Advisor 建議，請查詢 SVV_ALTER_TABLE_RECOMMENDATIONS 系統目錄檢視。如需詳細資訊，請參閱 SVV_ALTER_TABLE_RECOMMENDATIONS。若要檢視 Amazon Redshift 所採取的動作，請查詢 SVL_AUTO_WORKER_ACTION 系統目錄檢視。如需詳細資訊，請參閱 SVL_AUTO_WORKER_ACTION。

ALTER SORTKEY NONE

移除目標資料表的排序索引鍵的子句。

如果排序索引鍵先前已定義為 AUTO，則該資料表不再是自動資料表最佳化的候選項目。

ALTER ENCODE AUTO

將目標資料表資料欄的編碼類型變更為 AUTO 的子句。當您將編碼修改為 AUTO 時，Amazon Redshift 會保留資料表中資料欄的現有編碼類型。然後，如果 Amazon Redshift 判斷新的編碼類型可以改善查詢效能，Amazon Redshift 就可以變更資料表資料欄的編碼類型。

如果您修改一個或多個欄以指定編碼，則 Amazon Redshift 不會再自動調整資料表中所有資料欄的編碼。這些資料欄會保留目前的編碼設定。

下列動作不會影響資料表的 ENCODE AUTO 設定：

- 重新命名資料表。
- 修改資料表的 DISTSTYLE 或 SORTKEY 設定。
- 使用 ENCODE 設定新增或捨棄資料列。
- 使用 COPY 命令的 COMPUPDATE 選項。如需詳細資訊，請參閱 資料載入操作。

若要檢視資料表的編碼，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱 SVV_TABLE_INFO。

RENAME COLUMN column_name TO new_name

此子句會將資料欄重新命名為 new_name 中指定的值。資料欄名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

ADD [COLUMN] column_name

將指定名稱的資料欄新增至資料表的子句。您只能在每個 ALTER TABLE 陳述式中新增一個資料欄。

您無法新增本身為資料表的分佈索引鍵 (DISTKEY) 或排序索引鍵 (SORTKEY) 的資料欄。

您無法使用 ALTER TABLE ADD COLUMN 命令修改以下資料表和資料欄屬性：

- UNIQUE
- PRIMARY KEY
- REFERENCES (外部索引鍵)
- IDENTITY 或 GENERATED BY DEFAULT AS IDENTITY

資料欄名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。單一資料表中可定義的資料欄數目上限為 1,600 個。

以下限制適用於新增資料欄至外部資料表時：

- 您無法將資料欄新增至具有資料欄限制條件 DEFAULT、ENCODE、NOT NULL 或 NULL 的外部資料表。
- 您無法將資料欄新增至使用 AVRO 檔案格式定義的外部資料表。
- 若啟用虛擬資料欄，則單一外部資料表中可定義的資料欄數目上限為 1,598 個。若未啟用虛擬資料欄，則單一資料表中可定義的資料欄數目上限為 1,600 個。

如需詳細資訊，請參閱 CREATE EXTERNAL TABLE。

column_type

要新增之資料欄的資料類型。若是 CHAR 和 VARCHAR 資料欄，您可以改用 MAX 關鍵字，而不宣告長度上限。MAX 會將 CHAR 的長度上限設定為 4,096 個位元組，或將 VARCHAR 的長度上限設定為 65,535 個位元組。GEOMETRY 物件的大小上限是 1,048,447 位元組。

如需有關 Amazon Redshift 支援資料類型的資訊，請參閱 資料類型。

DEFAULT default_expr

此子句會指派資料欄的預設資料值。default_expr 的資料類型必須符合資料欄的資料類型。DEFAULT 值必須是無變數的表達式。不允許子查詢、目前資料表中其他資料欄的交叉參考，以及使用者定義的功能。

default_expr 是在未指定資料欄值的任何 INSERT 操作中使用。若未指定預設值，則資料欄的預設值為 null。

若 COPY 操作在資料欄上遇到有 DEFAULT 值和 NOT NULL 限制條件的 null 欄位，則 COPY 命令會插入 default_expr 的值。

外部資料表不支援 DEFAULT。

ENCODE encoding

資料欄的壓縮編碼。根據預設，如果您未為資料表中的任何資料欄指定壓縮編碼，或者您指定資料表的 ENCODE AUTO 選項，Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。

如果您為資料表中的任何資料欄指定壓縮編碼，或者未為資料表指定 ENCODE AUTO 選項，Amazon Redshift 會自動將壓縮編碼指派給您未指定壓縮編碼的資料欄，如下所示：

- 暫時資料表中的所有資料欄預設都會有指派的 RAW 壓縮。
- 定義為排序索引鍵的資料欄會有指派的 RAW 壓縮。
- 定義為 BOOLEAN、REAL、DOUBLE PRECISION、GEOMETRY 或 GEOGRAPHY 資料類型的資料行會有指派的 RAW 壓縮。

- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP 或 TIMESTAMPTZ 的資料欄會有指派的 AZ64 壓縮。
- 定義為 CHAR、VARCHAR 或 VARBYTE 的資料欄會有指派的 LZO 壓縮。

> **Note**
>
> 若您不想要壓縮資料欄，請明確指定 RAW 編碼。

支援以下 compression encodings (p. 55)：

- AZ64
- BYTEDICT
- DELTA
- DELTA32K
- LZO
- MOSTLY8
- MOSTLY16
- MOSTLY32
- RAW (無壓縮)
- RUNLENGTH
- TEXT255
- TEXT32K
- ZSTD

外部資料表不支援 ENCODE。

NOT NULL | NULL

NOT NULL 會指定不允許資料欄包含 null 值。NULL 是預設值，指出資料欄可接受 null 值。

外部資料表不支援 NOT NULL 和 NULL。

COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }

指定資料欄上的字串搜尋或比較是 CASE_SENSITIVE 或 CASE_INSENSITIVE 的子句。預設值與資料庫目前的區分大小寫設定相同。

若要尋找資料庫定序資訊，請使用下列命令：

```
SELECT db_collation();

db_collation
----------------
 case_sensitive
(1 row)
```

DROP [COLUMN] column_name

要從資料表中刪除的資料欄名稱。

您無法刪除資料表中的最後一個資料欄。資料表至少必須有一個資料欄。

您無法刪除本身為資料表的分佈索引鍵 (DISTKEY) 或排序索引鍵 (SORTKEY) 的資料欄。若資料欄有任何相依物件，像是檢視、主索引鍵、外部索引鍵或 UNIQUE 限制條件，則 DROP COLUMN 的預設行為是 RESTRICT。

以下限制適用於從外部資料表刪除資料欄時：

- 若資料欄做為分割區使用，則無法從外部資料表刪除該資料欄。
- 您無法從使用 AVRO 檔案格式定義的外部資料表中刪除資料欄。
- 外部資料表的 RESTRICT 和 CASCADE 會遭到忽略。
- 除非您將政策捨棄或中斷連結，否則您無法捨棄政策定義中參照的政策資料表資料欄。這也適用於指定 CASCADE 選項時。您可以捨棄政策資料表中的其他資料欄。

如需詳細資訊，請參閱 CREATE EXTERNAL TABLE。

RESTRICT

搭配 DROP COLUMN 使用時，RESTRICT 表示要捨棄的資料欄未捨棄，在以下這些情況中：

- 如果定義的檢視參考要捨棄的資料欄
- 如果外部索引鍵參考資料欄
- 如果資料欄為分段索引鍵的一部分

RESTRICT 不能搭配 CASCADE 使用。

外部資料表的 RESTRICT 和 CASCADE 會遭到忽略。

CASCADE

搭配 DROP COLUMN 使用時，會移除指定的資料欄及相依於該資料欄的任何項目。CASCADE 不能搭配 RESTRICT 使用。

外部資料表的 RESTRICT 和 CASCADE 會遭到忽略。

以下選項只適用於外部資料表。

SET LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file' }

包含資料檔案的 Amazon S3 資料夾路徑，或包含 Amazon S3 物件路徑清單的資訊清單檔案。儲存貯體必須位於 Amazon Redshift 叢集所在的同一 AWS 區域。如需支援的 AWS 區域清單，請參閱Amazon Redshift Spectrum 考量事項。如需使用資訊清單檔案的相關資訊，請參閱 CREATE EXTERNAL TABLE 參數 參考中的 LOCATION。

SET FILE FORMAT format

外部資料檔案的檔案格式。

有效格式如下：

- AVRO
- PARQUET
- RCFILE
- SEQUENCEFILE
- TEXTFILE

SET TABLE PROPERTIES ('property_name'='property_value')

此子句會設定外部資料表的資料表屬性的資料表定義。

> **Note**
>
> 資料表屬性區分大小寫。

'numRows'='row_count'

此屬性會設定資料表定義的 numRows 值。若要明確更新外部資料表的統計資料，請設定 numRows 屬性以指出資料表的大小。Amazon Redshift 不會分析外部資料表來產生查詢最佳化工具用來產生查詢計劃的資料表統計資料。如果未設定外部資料表的資料表統計資料，則 Amazon Redshift 會產生查詢執行計畫。此計畫是根據外部資料表為較大資料表，而本機資料表為較小資料表的假設。

'skip.header.line.count'='line_count'

此屬性會設定每個來源檔案開頭要略過的資料列數。

PARTITION (partition_column=partition_value [, ...] SET LOCATION { 's3://bucket/folder' | 's3://bucket/manifest_file' }

此子句會設定一個或多個分割區資料欄的新位置。

ADD [IF NOT EXISTS] PARTITION (partition_column=partition_value [, ...]) LOCATION { 's3://bucket/folder' | 's3://bucket/manifest_file' } [, ...]

新增一或多個分割區的子句。您可以使用單一 ALTER TABLE … ADD 陳述式來指定多個 PARTITION 子句。

> **Note**
>
> 如果您使用目 AWS Glue 錄，則可以使用單一 ALTER TABLE 陳述式新增最多 100 個分割區。

IF NOT EXISTS 子句指出，若指定的分割區已存在，則命令不應進行任何變更。也指出命令應該傳回分割區已存在的訊息，而不是在發生錯誤的情況下終止。此子句在編寫指令碼時很實用，如此指令碼就不會因為 ALTER TABLE 嘗試新增已存在的分割區而失敗。

DROP PARTITION (partition_column=partition_value [, ...])

刪除指定分割區的子句。刪除分割區只會修改外部資料表中繼資料。Amazon S3 上的資料不受影響。

ROW LEVEL SECURITY { ON | OFF } [CONJUNCTION TYPE { AND | OR }] [FOR DATASHARES]

開啟或關閉關係的資料列層級安全性的子句。

開啟關係的資料列層級安全性時，您只能讀取資料列層級安全政策允許您存取的資料列。如果沒有任何政策授予您關係存取權，您就無法看到關聯中的任何資料列。只有超級使用者和具有 `sys:secadmin` 角色的使用者或角色可以設定 ROW LEVEL SECURITY 子句。如需詳細資訊，請參閱 資料列層級安全性。

- [CONJUNCTION TYPE { AND | OR }]

  一個子句，讓您為關係選擇資料列層級安全政策的結合類型。將多個資料列層級安全政策附加至關係時，您可以將政策與 AND 或 OR 子句結合使用。根據預設，Amazon Redshift 將 RLS 政策

與 AND 子句結合在一起。具有 sys:secadmin 角色的超級使用者、使用者或角色可以使用此子句來定義關係之資料列層級安全政策的組合類型。如需詳細資訊，請參閱 每個使用者結合多個政策。

- FOR DATASHARES

  該子句會決定是否可透過資料共用存取受 RLS 保護的關係。預設的情況是無法透過資料共用存取受 RLS 保護的關係。搭配此子句執行的 ALTER TABLE ROW LEVEL SECURITY 命令只會影響關係的資料共用存取屬性。ROW LEVEL SECURITY 屬性並未變更。

  如果您讓受 RLS 保護的關係可以透過資料庫存取，則該關係在取用者端資料庫中不會有資料列層級的安全性。關係會在生產者端保留其 RLS 屬性。

範例

如需示範如何使用 ALTER TABLE 命令的範例，請參閱下列內容。

- ALTER TABLE 範例
- ALTER EXTERNAL TABLE 範例
- ALTER TABLE ADD 和 DROP COLUMN 範例

ALTER TABLE 範例

以下範例示範 ALTER TABLE 命令的基本用法。

重新命名資料表或檢視

以下命令會將 USERS 資料表重新命名為 USERS_BKUP：

```
alter table users
rename to users_bkup;
```

您也可以使用此類型的命令來重新命名檢視。

變更資料表或檢視的擁有者

以下命令會將 VENUE 資料表擁有者變更為使用者 DWUSER：

```
alter table venue
```

```
owner to dwuser;
```

以下命令會建立檢視，然後變更其擁有者：

```
create view vdate as select * from date;
alter table vdate owner to vuser;
```

重新命名欄位

以下命令會將 VENUE 資料表中的 VENUESEATS 資料欄重新命名為 VENUESIZE：

```
alter table venue
rename column venueseats to venuesize;
```

丟棄資料表限制條件

若要捨棄資料表限制條件，像是主索引鍵、外部索引鍵或唯一限制條件，請先找到限制條件的內部名稱。然後在 ALTER TABLE 命令中指定限制條件名稱。以下範例會尋找 CATEGORY 資料表的限制條件，然後刪除名稱為 category_pkey 的主索引鍵。

```
select constraint_name, constraint_type
from information_schema.table_constraints
where constraint_schema ='public'
and table_name = 'category';

constraint_name | constraint_type
----------------+----------------
category_pkey   | PRIMARY KEY

alter table category
drop constraint category_pkey;
```

變更 VARCHAR 資料欄

若要節省儲存體，您可以定義一個資料表，其中的 VARCHAR 資料欄一開始具有您目前資料需求所需的最低大小。之後若要容納較長的字串，則可以變更資料表以增加資料欄的大小。

以下範例會將 EVENTNAME 資料欄的大小增加為 VARCHAR (300)。

```
alter table event alter column eventname type varchar(300);
```

修改資料欄的壓縮編碼。

您可以修改資料欄的壓縮編碼。您可以在下方找到一組示範此方法的範例。這些範例的資料表定義如下。

```
create table t1(c0 int encode lzo, c1 bigint encode zstd, c2 varchar(16) encode lzo, c3
 varchar(32) encode zstd);
```

下列陳述式會將資料欄 c0 的壓縮編碼從 LZO 編碼修改為 AZ64 編碼。

```
alter table t1 alter column c0 encode az64;
```

下列陳述式會將資料欄 c1 的壓縮編碼從 Zstandard 編碼修改為 AZ64 編碼。

```
alter table t1 alter column c1 encode az64;
```

下列陳述式會將資料欄 c2 的壓縮編碼從 LZO 編碼修改為 Byte-Dictionary 編碼。

```
alter table t1 alter column c2 encode bytedict;
```

下列陳述式會將資料欄 c3 的壓縮編碼從 Zstandard 編碼修改為 Runlength 編碼。

```
alter table t1 alter column c3 encode runlength;
```

修改 DISTSTYLE KEY DISTKEY 欄位

下列範例示範如何變更資料表的 DISTSTYLE 和 DISTKEY。

建立採用 EVEN 分佈樣式的資料表。SVV_TABLE_INFO 檢視顯示 DISTSTYLE 為 EVEN。

```
create table inventory(
  inv_date_sk int4 not null ,
  inv_item_sk int4 not null ,
  inv_warehouse_sk int4 not null ,
  inv_quantity_on_hand int4
) diststyle even;

Insert into inventory values(1,1,1,1);
```

```
select "table", "diststyle" from svv_table_info;

   table   |   diststyle
-----------+---------------
 inventory |     EVEN
```

將資料表 DISTKEY 改為 inv_warehouse_sk。SVV_TABLE_INFO 檢視將 inv_warehouse_sk 資料欄顯示為產生的分佈金鑰。

```
alter table inventory alter diststyle key distkey inv_warehouse_sk;

select "table", "diststyle" from svv_table_info;

   table   |       diststyle
-----------+-----------------------
 inventory | KEY(inv_warehouse_sk)
```

將資料表 DISTKEY 改為 inv_item_sk。SVV_TABLE_INFO 檢視將 inv_item_sk 資料欄顯示為產生的分佈金鑰。

```
alter table inventory alter distkey inv_item_sk;

select "table", "diststyle" from svv_table_info;

   table   |       diststyle
-----------+-----------------------
 inventory | KEY(inv_item_sk)
```

將資料表改為 DISTSTYLE ALL

下列範例示範如何將資料表變更為 DISTSTYLE ALL。

建立採用 EVEN 分佈樣式的資料表。SVV_TABLE_INFO 檢視顯示 DISTSTYLE 為 EVEN。

```
create table inventory(
  inv_date_sk int4 not null ,
  inv_item_sk int4 not null ,
  inv_warehouse_sk int4 not null ,
  inv_quantity_on_hand int4
) diststyle even;
```

```
Insert into inventory values(1,1,1,1);

select "table", "diststyle" from svv_table_info;

   table   |   diststyle
-----------+----------------
 inventory |     EVEN
```

將資料表 DISTSTYLE 改為 ALL。SVV_TABLE_INFO 檢視會顯示已變更的 DISTSYTLE。

```
alter table inventory alter diststyle all;

select "table", "diststyle" from svv_table_info;

   table   |   diststyle
-----------+----------------
 inventory |     ALL
```

修改資料表 SORTKEY

您可以修改資料表以具有複合排序索引鍵或沒有排序鍵。

在下列資料表定義中，資料表 t1 是以交錯排序索引鍵來定義。

```
create table t1 (c0 int, c1 int) interleaved sortkey(c0, c1);
```

下列命令會將資料表從交錯排序索引鍵修改為複合排序索引鍵。

```
alter table t1 alter sortkey(c0, c1);
```

下列命令會修改資料表以移除交錯排序索引鍵。

```
alter table t1 alter sortkey none;
```

在下列資料表定義中，資料表 t1 是以 c0 作為交錯排序索引鍵來定義。

```
create table t1 (c0 int, c1 int) sortkey(c0);
```

下列命令會將資料表 t1 修改為複合排序索引鍵。

```
alter table t1 alter sortkey(c0, c1);
```

將資料表修改為 ENCODE AUTO

下列範例顯示如何將資料表修改為 ENCODE AUTO。

此範例的資料表定義如下。資料欄 c0 是以編碼類型 AZ64 定義，而資料欄 c1 是以編碼類型 LZO 定義。

```
create table t1(c0 int encode AZ64, c1 varchar encode LZO);
```

對於此資料表，下列陳述式會將編碼變更為 AUTO。

```
alter table t1 alter encode auto;
```

下列範例會顯示如何將資料表修改為移除 ENCODE AUTO 設定。

此範例的資料表定義如下。資料表資料欄不需編碼即可定義。在這種情況下，編碼會預設為 ENCODE AUTO。

```
create table t2(c0 int, c1 varchar);
```

對於此資料表，下列陳述式會將資料欄 c0 的編碼變更為 LZO。資料表編碼不再設定為 ENCODE AUTO。

```
alter table t2 alter column c0 encode lzo;;
```

修改資料列層級安全性控制

下列命令會關閉資料表的 RLS：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY OFF;
```

下列命令會開啟資料表的 RLS：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;
```

下列命令會開啟資料表的 RLS，並使其可透過資料共用存取：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY FOR DATASHARES OFF;
```

下列命令會開啟資料表的 RLS，並使其不可透過資料共用存取：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY FOR DATASHARES ON;
```

下列命令會開啟 RLS，並將資料表的 RLS 結合類型設定為 OR：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON CONJUNCTION TYPE OR;
```

下列命令會開啟 RLS，並將資料表的 RLS 結合類型設定為 AND：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON CONJUNCTION TYPE AND;
```

ALTER EXTERNAL TABLE 範例

下列範例使用位於美國東部 (維吉尼亞北部) 區域 (us-east-1) 的 Amazon S3 儲存貯體，以 AWS 區域 及範例針對 CREATE TABLE 建立的範例表格。如需如何搭配外部資料表使用分割區的詳細資訊，請參閱分割 Redshift Spectrum 外部資料表。

以下範例會將 SPECTRUM.SALES 外部資料表的 numRows 資料表屬性設定為 170,000 個資料列。

```
alter table spectrum.sales
set table properties ('numRows'='170000');
```

以下範例會變更 SPECTRUM.SALES 外部資料表的位置。

```
alter table spectrum.sales
set location 's3://redshift-downloads/tickit/spectrum/sales/';
```

以下範例會將 SPECTRUM.SALES 外部資料表的格式變更為 Parquet。

```
alter table spectrum.sales
set file format parquet;
```

以下範例會為 SPECTRUM.SALES_PART 資料表新增一個分割區。

```
alter table spectrum.sales_part
add if not exists partition(saledate='2008-01-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/';
```

以下範例會為 SPECTRUM.SALES_PART 資料表新增三個分割區。

```
alter table spectrum.sales_part add if not exists
partition(saledate='2008-01-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/'
partition(saledate='2008-02-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/'
partition(saledate='2008-03-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/';
```

以下範例會修改 SPECTRUM.SALES_PART 以刪除具有 saledate='2008-01-01'' 的分割區。

```
alter table spectrum.sales_part
drop partition(saledate='2008-01-01');
```

以下範例會為具有 saledate='2008-01-01' 的分割區設定新的 Amazon S3 路徑。

```
alter table spectrum.sales_part
partition(saledate='2008-01-01')
set location 's3://redshift-downloads/tickit/spectrum/sales_partition/
saledate=2008-01-01/';
```

下列範例會將 sales_date 的名稱變更為 transaction_date。

```
alter table spectrum.sales rename column sales_date to transaction_date;
```

下列範例會針對使用最佳化資料列單欄式 (ORC) 格式的外部資料表，將資料欄映射設定為位置映射。

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='position');
```

下列範例會針對使用 ORC 格式的外部資料表，將資料欄映射設定為名稱映射。

```
alter table spectrum.orc_example
```

```
set table properties('orc.schema.resolution'='name');
```

ALTER TABLE ADD 和 DROP COLUMN 範例

以下範例示範如何使用 ALTER TABLE 先新增再刪除基本資料表資料欄，以及如何刪除具有相依物件的資料欄。

先 ADD 再 DROP 基本欄位

下列範例會將獨立的 FEEDBACK_SCORE 資料欄新增至 USERS 資料表。此資料欄僅包含整數，且此資料欄的預設值為 NULL (無意見回饋分數)。

首先，查詢 PG_TABLE_DEF 目錄資料表以檢視 USERS 資料表的結構描述：

```
column        | type                    | encoding | distkey | sortkey
--------------+-------------------------+----------+---------+--------
userid        | integer                 | delta    | true    |       1
username      | character(8)            | lzo      | false   |       0
firstname     | character varying(30)   | text32k  | false   |       0
lastname      | character varying(30)   | text32k  | false   |       0
city          | character varying(30)   | text32k  | false   |       0
state         | character(2)            | bytedict | false   |       0
email         | character varying(100)  | lzo      | false   |       0
phone         | character(14)           | lzo      | false   |       0
likesports    | boolean                 | none     | false   |       0
liketheatre   | boolean                 | none     | false   |       0
likeconcerts  | boolean                 | none     | false   |       0
likejazz      | boolean                 | none     | false   |       0
likeclassical | boolean                 | none     | false   |       0
likeopera     | boolean                 | none     | false   |       0
likerock      | boolean                 | none     | false   |       0
likevegas     | boolean                 | none     | false   |       0
likebroadway  | boolean                 | none     | false   |       0
likemusicals  | boolean                 | none     | false   |       0
```

現在新增 feedback_score 資料欄：

```
alter table users
add column feedback_score int
default NULL;
```

從 USERS 選取 FEEDBACK_SCORE 資料欄，確認其已新增：

```
select feedback_score from users limit 5;

feedback_score
----------------
NULL
NULL
NULL
NULL
NULL
```

刪除資料欄以恢復原始 DDL：

```
alter table users drop column feedback_score;
```

捨棄具有相依物件的欄位

下列範例會捨棄具有相依物件的資料欄。結果會將相依物件一併刪除。

首先再次將 FEEDBACK_SCORE 資料欄新增至 USERS 資料表：

```
alter table users
add column feedback_score int
default NULL;
```

接著從 USERS 資料表建立名為 USERS_VIEW 的檢視：

```
create view users_view as select * from users;
```

現在嘗試從 USERS 資料表刪除 FEEDBACK_SCORE 資料欄。此 DROP 陳述式會使用預設行為 (RESTRICT)：

```
alter table users drop column feedback_score;
```

Amazon Redshift 會顯示錯誤訊息，指出無法捨棄資料欄，因為有其他物件與其相依。

再次嘗試刪除 FEEDBACK_SCORE 資料欄，這次指定 CASCADE 以刪除所有相依物件：

```
alter table users
drop column feedback_score cascade;
```

ALTER TABLE APPEND

會從現有來源資料表移出資料，將資料列附加到目標資料表。來源資料表中的資料會移至目標資料表中相符的資料欄。資料欄的順序並不重要。資料成功附加至目標資料表後，來源資料表就會是空的。因為是移動而不是複製資料，ALTER TABLE APPEND 通常比類似的 CREATE TABLE AS 或 INSERT INTO 操作更快。

> **Note**
>
> ALTER TABLE APPEND 會在來源資料表和目標資料表之間移動資料區塊。為了提升效能，ALTER TABLE APPEND 不會在附加操作的過程中精簡儲存空間。因此，儲存空間的使用量會暫時增加。若要回收空間，請執行 VACUUM 操作。

同名的資料欄也必須有一模一樣的資料欄屬性。若來源資料表或目標資料表包含對方所沒有的資料欄，請使用 IGNOREEXTRA 或 FILLTARGET 參數指定管理額外資料欄的方式。

您無法附加身分資料欄。若兩個資料表都包含身分資料欄，命令就會失敗。若只有一個資料表擁有身分資料欄，請包含 FILLTARGET 或 IGNOREEXTRA 參數。如需詳細資訊，請參閱 ALTER TABLE APPEND 使用須知。

您可以附加 GENERATED BY DEFAULT AS IDENTITY 欄。您可以利用您提供的值，更新定義為 GENERATED BY DEFAULT AS IDENTITY 的資料欄。如需詳細資訊，請參閱 ALTER TABLE APPEND 使用須知。

目標資料表必須是永久資料表。不過，來源可以是永久資料表，也可以是為串流擷取設定的具體化視觀表。兩個物件必須使用相同的分佈樣式和分佈索引鍵 (如有定義的話)。若物件經過排序，則兩個物件必須使用相同的排序樣式，並且定義相同的資料欄做為排序索引鍵。

ALTER TABLE APPEND 命令會在操作完成時立即自動遞交。此命令無法轉返。您無法在交易區塊 (BEGIN ... END) 內執行 ALTER TABLE APPEND。如需交易的相關資訊，請參閱 可序列化隔離。

所需權限

根據 ALTER TABLE APPEND 命令的不同，使用者可能需要下列其中一項權限：

- 超級使用者
- 具 ALTER TABLE 系統權限的使用者
- 在來源資料表上具有 DELETE 和 SELECT 權限，以及在目標資料表上具有 INSERT 權限的使用者

語法

```
ALTER TABLE target_table_name APPEND FROM [ source_table_name
 | source_materialized_view_name ]
[ IGNOREEXTRA | FILLTARGET ]
```

從具體化視觀表附加只有在為 串流擷取 設定具體化視觀表的情況下才有作用。

參數

target_table_name

要附加資料列的資料表名稱。僅指定資料表的名稱，或使用格式 schema_name.table_name 來使用專屬結構描述。目標資料表必須是現有的永久資料表。

FROM source_table_name

提供要附加之資料列的資料表名稱。僅指定資料表的名稱，或使用格式 schema_name.table_name 來使用專屬結構描述。來源資料表必須是現有的永久資料表。

FROM source_materialized_view_name

提供要附加之資料列的具體化視觀表名稱。從具體化視觀表附加只有在為 串流擷取 設定具體化視觀表的情況下才有作用。來源具體化視觀表必須已存在。

IGNOREEXTRA

此關鍵字會指定，若來源資料表包含了目標資料表中沒有的資料欄，則應捨棄額外資料欄中的資料。您無法搭配 FILLTARGET 使用 IGNOREEXTRA。

FILLTARGET

此關鍵字會指定，若目標資料表包含了來源資料表中沒有的資料欄，則應在這些資料欄中填入 DEFAULT 資料欄值 (如有定義的話) 或 NULL。您無法搭配 FILLTARGET 使用 IGNOREEXTRA。

ALTER TABLE APPEND 使用須知

ALTER TABLE APPEND 只會移動來源資料表與目標資料表中完全相同的資料欄。資料欄的順序並不重要。

若來源資料表或目標資料表包含額外的資料欄，請根據下列規則使用 FILLTARGET 或 IGNOREEXTRA：

- 如果來源資料表包含了目標資料表中沒有的資料欄，則要包括 IGNOREEXTRA。此命令會略過來源資料表中額外的資料欄。
- 如果目標資料表包含了來源資料表中沒有的資料欄，則要包括 FILLTARGET。此命令會以預設資料欄值或 IDENTITY 值 (如有定義的話) 或 NULL 填入目標資料表中額外的資料欄。
- 如果來源資料表和目標資料表都包含額外的資料欄，則命令會失敗。您無法同時使用 FILLTARGET 和 IGNOREEXTRA。

如果兩個資料表中包含的資料欄名稱相同但屬性不同，則命令會失敗。名稱類似的資料欄必須都有下列屬性：

- 資料類型
- 資料欄大小
- 壓縮編碼
- 非 null
- 排序樣式
- 排序索引鍵資料欄
- 分佈樣式
- 分佈索引鍵資料欄

您無法附加身分資料欄。如果來源資料表和目標資料表都有身分資料欄，則命令會失敗。若只有來源資料表擁有身分資料欄，請包含 IGNOREEXTRA 參數以便略過身分資料欄。若只有目標資料表擁有身分資料欄，請包含 FILLTARGET 參數，以便根據為資料表定義的 IDENTITY 子句填入身分資料欄。如需詳細資訊，請參閱 DEFAULT。

您可以利用 ALTER TABLE APPEND 陳述式來附加預設身分資料欄。如需詳細資訊，請參閱 CREATE TABLE。

ALTER TABLE APPEND 範例

假設您的組織有資料表 SALES_MONTHLY，用來擷取目前的銷售交易。您想要每個月將交易資料表中的資料移至 SALES 資料表。

您可以使用以下 INSERT INTO 和 TRUNCATE 命令完成這項任務。

```
insert into sales (select * from sales_monthly);
```

```
truncate sales_monthly;
```

不過，使用 ALTER TABLE APPEND 命令能夠更有效率地執行相同的操作。

首先，查詢 PG_TABLE_DEF 系統目錄資料表以確認兩個資料表擁有相同的資料欄，且資料欄的屬性相同。

```
select trim(tablename) as table, "column", trim(type) as type,
encoding, distkey, sortkey, "notnull"
from pg_table_def where tablename like 'sales%';

table      | column     | type                          | encoding | distkey | sortkey |
 notnull
-----------+------------+-------------------------------+----------+---------+---------
+--------
sales      | salesid    | integer                       | lzo      | false   |       0 |
 true
sales      | listid     | integer                       | none     | true    |       1 |
 true
sales      | sellerid   | integer                       | none     | false   |       2 |
 true
sales      | buyerid    | integer                       | lzo      | false   |       0 |
 true
sales      | eventid    | integer                       | mostly16 | false   |       0 |
 true
sales      | dateid     | smallint                      | lzo      | false   |       0 |
 true
sales      | qtysold    | smallint                      | mostly8  | false   |       0 |
 true
sales      | pricepaid  | numeric(8,2)                  | delta32k | false   |       0 |
 false
sales      | commission | numeric(8,2)                  | delta32k | false   |       0 |
 false
sales      | saletime   | timestamp without time zone | lzo      | false   |       0 |
 false
salesmonth | salesid    | integer                       | lzo      | false   |       0 |
 true
salesmonth | listid     | integer                       | none     | true    |       1 |
 true
salesmonth | sellerid   | integer                       | none     | false   |       2 |
 true
salesmonth | buyerid    | integer                       | lzo      | false   |       0 |
 true
```

```
salesmonth | eventid    | integer                     | mostly16 | false   |       0 |
 true
salesmonth | dateid     | smallint                    | lzo      | false   |       0 |
 true
salesmonth | qtysold    | smallint                    | mostly8  | false   |       0 |
 true
salesmonth | pricepaid  | numeric(8,2)                | delta32k | false   |       0 |
 false
salesmonth | commission | numeric(8,2)                | delta32k | false   |       0 |
 false
salesmonth | saletime   | timestamp without time zone | lzo      | false   |       0 |
 false
```

接著查看各資料表的大小。

```
select count(*) from sales_monthly;
 count
-------
  2000
(1 row)

select count(*) from sales;
 count
-------
 412,214
(1 row)
```

現在執行以下 ALTER TABLE APPEND 命令。

```
alter table sales append from sales_monthly;
```

再次查看各資料表的大小。SALES_MONTHLY 資料表現在擁有 0 個資料列，而 SALES 資料表增加了 2000 個資料列。

```
select count(*) from sales_monthly;
 count
-------
     0
(1 row)

select count(*) from sales;
 count
```

```
-------
 414214
(1 row)
```

如果來源資料表的資料欄比目標資料表多，請指定 IGNOREEXTRA 參數。以下範例使用 IGNOREEXTRA 參數在附加至 SALES 資料表時，略過 SALES_LISTING 資料表中額外的資料欄。

```
alter table sales append from sales_listing ignoreextra;
```

如果目標資料表的資料欄比來源資料表多，請指定 FILLTARGET 參數。以下範例使用 FILLTARGET 參數來填入 SALES_REPORT 資料表中 SALES_MONTH 資料表沒有的資料欄。

```
alter table sales_report append from sales_month filltarget;
```

下列範例顯示如何以具體化視觀表作為來源來使用 ALTER TABLE APPEND 的範例。

```
ALTER TABLE target_tbl APPEND FROM my_streaming_materialized_view;
```

此範例中的資料表和具體化視觀表名稱為範例。從具體化視觀表附加只有在為 串流擷取 設定具體化視觀表的情況下才有作用。這會將來源具體化視觀表中的所有記錄移至與具體化視觀表具有相同結構描述的目標資料表，並使具體化視觀表保持不變。這與資料來源為資料表時的行為相同。

ALTER USER

變更資料庫使用者。

所需權限

以下是 ALTER USER 所需的權限：

- 超級使用者
- 具有 ALTER USER 權限的使用者
- 想要變更自己密碼的目前使用者

語法

```
ALTER USER username [ WITH ] option [, ... ]
```

```
where option is

CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }
| PASSWORD { 'password' | 'md5hash' | DISABLE }
[ VALID UNTIL 'expiration_date' ]
| RENAME TO new_name |
| CONNECTION LIMIT { limit | UNLIMITED }
| SESSION TIMEOUT limit | RESET SESSION TIMEOUT
| SET parameter { TO | = } { value | DEFAULT }
| RESET parameter
| EXTERNALID external_id
```

參數

username

使用者的名稱。

WITH

選用的關鍵字。

CREATEDB | NOCREATEDB

CREATEDB 選項可讓使用者建立新的資料庫。NOCREATEDB 是預設值。

CREATEUSER | NOCREATEUSER

CREATEUSER 選項可建立具有所有資料庫權限的超級使用者，包括 CREATE USER。預設值為 NOCREATEUSER。如需詳細資訊，請參閱 superuser。

SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }

此子句會指定使用者對 Amazon Redshift 系統資料表和檢視的存取層級。

具有 SYSLOG ACCESS 受限制權限的一般使用者只能在使用者可見的系統資料表和檢視中看到該使用者所產生的資料列。預設值為 RESTRICTED。

具有 SYSLOG ACCESS UNCONCTED 權限的一般使用者可以看到使用者可見系統資料表和檢視中的所有資料列，包括其他使用者產生的資料列。UNRESTRICTED 並不會讓一般使用者存取超級使用者可查看的資料表。只有超級使用者可看見超級使用者可查看的資料表。

> **Note**
>
> 若使用者擁有不受限制的系統資料表存取權限，該使用者就能查看其他使用者產生的資料。例如，STL_QUERY 和 STL_QUERYTEXT 包含 INSERT、UPDATE 和 DELETE 陳述式的全文，當中可能包含使用者產生的敏感資料。

所有使用者皆可看到 SVV_TRANSACTIONS 中的所有資料列。

如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

PASSWORD { 'password' | 'md5hash' | DISABLE }

設定使用者的密碼。

根據預設，使用者可以變更自己擁有的密碼，除非密碼遭到停用。若要停用使用者的密碼，請指定 DISABLE。停用使用者密碼時，密碼會從系統中刪除，使用者只能使用臨時 AWS Identity and Access Management (IAM) 使用者登入資料登入。如需詳細資訊，請參閱使用 IAM 身分驗證產生資料庫使用者登入資料。只有超級使用者能夠啟用或停用密碼，您無法停用超級使用者的密碼。若要啟用密碼，請執行 ALTER USER 並指定密碼。

如需使用 PASSWORD 參數的詳細資訊，請參閱 CREATE USER。

VALID UNTIL 'expiration_date'

指定密碼有過期日期。使用 `'infinity'` 值就不會有過期日期。此參數的有效資料類型為時間戳記。

只有超級使用者可以使用此參數。

RENAME TO

重新命名使用者。

new_name

使用者的新名稱。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

> **Important**
>
> 重新命名使用者時，您也必須重設使用者的密碼。重設密碼不一定要與先前的密碼不同。使用者名稱會做為密碼加密的一部分使用，因此使用者重新命名時，密碼也會遭到清除。在密碼重設前，使用者將無法登入。例如：

```
alter user newuser password 'EXAMPLENewPassword11';
```

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字可允許同時連線的最大數目。另外也可能限制每個資料庫的連線數目。如需詳細資訊，請參閱 CREATE DATABASE。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 STV_SESSIONS 系統畫面。

> **Note**
>
> 如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

SESSION TIMEOUT limit | RESET SESSION TIMEOUT

工作階段保持非作用中或閒置的時間上限 (以秒為單位)。範圍是 60 秒 (一分鐘) 到 1,728,000 秒 (20 天)。如果未為使用者設定工作階段逾時，則會套用叢集設定。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制。

當您設定工作階段逾時時，只會套用至新的工作階段。

若要檢視有關作用中使用者工作階段的資訊 (包括開始時間、使用者名稱和工作階段逾時)，請查詢 STV_SESSIONS 系統檢視。若要檢視有關使用者工作階段歷史記錄的資訊，請查詢 STL_SESSIONS 檢視。若要擷取有關資料庫使用者的資訊 (包括工作階段逾時值)，請查詢 SVL_USER_INFO 檢視。

SET

將指定使用者執行之所有工作階段的組態參數設定為新的預設值。

RESET

將指定使用者的組態參數重設為原始預設值。

parameter

要設定或重設的參數名稱。

值

參數的新值。

DEFAULT

將指定使用者執行之所有工作階段的組態參數設定為預設值。

EXTERNALID external_id

與身分提供者相關聯的使用者識別碼。使用者必須停用密碼。如需詳細資訊，請參閱 Amazon Redshift 的原生身分提供者 (IdP) 聯合合。

使用須知

- 嘗試修改 rdsdb - 您不能修改名為 `rdsdb` 的使用者。
- 建立未知密碼 — 使用 AWS Identity and Access Management (IAM) 身份驗證建立資料庫使用者登入資料時，您可能想要建立僅能使用臨時登入資料登入的超級使用者。您無法停用超級使用者的密碼，但是可以使用隨機產生的 MD5 雜湊字串建立未知的密碼。

```
alter user iam_superuser password 'md5123456789012345678012345678901 2';
```

- 設定 search_path – 當您使用 ALTER USER 命令設定 search_path 參數時，修改會在指定使用者下次登入時生效。若您想要變更目前使用者和工作階段的 search_path 值，請使用 SET 命令。
- 設定時區 – 當您搭配 ALTER USER 命令使用 SET TIMEZONE 時，修改會在指定使用者下次登入時生效。
- 使用動態資料遮罩和資料列層級安全性政策 — 當您佈建的叢集或無伺服器命名空間具有任何動態資料遮罩或資料列層級安全性政策時，一般使用者會無法使用下列命令：

```
ALTER <current_user> SET enable_case_sensitive_super_attribute/
enable_case_sensitive_identifier/downcase_delimited_identifier
```

只有超級使用者和具有 ALTER USER 權限的使用者才能設定這些組態選項。如需有關資料列層級安全性詳細資訊，請參閱 資料列層級安全性。如需動態資料遮罩的詳細資訊，請參閱 動態資料遮罩。

範例

下列範例會將建立資料庫的權限提供給使用者 ADMIN：

```
alter user admin createdb;
```

下面範例會將使用者 ADMIN 的密碼設為 adminPass9，以及為密碼設定過期日期和時間：

```
alter user admin password 'adminPass9'
valid until '2017-12-31 23:59';
```

以下範例會將使用者 ADMIN 重新命名為 SYSADMIN：

```
alter user admin rename to sysadmin;
```

下列範例會將使用者的閒置工作階段逾時更新為 300 秒。

```
ALTER USER dbuser SESSION TIMEOUT 300;
```

重設使用者的閒置工作階段逾時。重設時會套用叢集設定。您必須是資料庫超級使用者才能執行此命令。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制。

```
ALTER USER dbuser RESET SESSION TIMEOUT;
```

下列範例會更新名為 bob 之使用者的外部 ID。命名空間為 myco_aad。如果命名空間未與已註冊的身分提供者相關聯，則會導致錯誤。

```
ALTER USER myco_aad:bob EXTERNALID "ABC123" PASSWORD DISABLE;
```

下列範例會為特定資料庫使用者執行的所有工作階段設定時區。這會變更後續工作階段的時區，但不會變更目前工作階段的時區。

```
ALTER USER odie SET TIMEZONE TO 'Europe/Zurich';
```

下列範例會設定允許使用者bob開啟的資料庫連線數目上限。

```
ALTER USER bob CONNECTION LIMIT 10;
```

ANALYZE

更新資料表統計資訊以供查詢規劃器使用。

所需權限

以下是 ANALYZE 所需的權限：

- 超級使用者
- 具有 ANALYZE 權限的使用者
- 關係的擁有者
- 共用資料表的資料庫擁有者

語法

```
ANALYZE [ VERBOSE ]
[ [ table_name [ ( column_name [, ...] ) ] ]
[ PREDICATE COLUMNS | ALL  COLUMNS ]
```

參數

詳細

此子句會傳回有關 ANALYZE 操作的進度資訊訊息。若您不指定資料表，此選項會很實用。

table_name

您可以分析特定資料表，包括臨時資料表。您可以利用結構描述名稱限定資料表。您可以選擇性的指定 table_name 來分析單一資料表。您無法在單一 ANALYZE table_name 陳述式中指定多個 table_name。如果您未指定 table_name 值，則目前連接的資料庫中所有資料表都會加以分析，包括系統目錄中的永久性資料表。如果自上次 ANALYZE 之後變更的資料列百分比低於分析閾值，Amazon Redshift 會略過分析資料表。如需詳細資訊，請參閱 分析閾值。

您不需要分析 Amazon Redshift 系統資料表 (STL 和 STV 資料表)。

column_name

若您指定 table_name，則您也可以在資料表中指定一個或多個資料欄 (括號內以資料欄分隔的清單)。若指定了資料欄清單，則只會分析列出的資料欄。

PREDICATE COLUMNS | ALL COLUMNS

這些子句會指出，ANALYZE 是否應該只包含述詞資料欄。指定 PREDICATE COLUMNS 時，只會分析已在先前查詢中做為述詞使用的資料欄，或可能做為述詞使用的候選資料欄。指定 ALL COLUMNS 則會分析所有資料欄。預設值是 ALL COLUMNS。

若下列任一條件為真，表示資料欄包含在一組述詞資料欄中：

- 資料欄已在查詢中做為篩選的一部分、聯結條件或 group by 子句使用。
- 資料欄是分佈索引鍵。
- 資料欄是排序索引鍵的一部分。

如果資料表尚未經過查詢，因而未將任何資料欄標示為述詞資料欄，則即使指定了 PREDICATE COLUMNS，仍會分析所有資料欄。如需述詞資料欄的相關資訊，請參閱 分析資料表。

使用須知

Amazon Redshift 會在您使用下列命令建立的資料表上自動執行 ANALYZE：

- CREATE TABLE AS
- CREATE TEMP TABLE AS
- SELECT INTO

您無法分析外部資料表。

您不需要在初次建立這些資料表時，對其執行 ANALYZE 命令。若您修改這些資料表，則應依照與其他資料表相同的方式進行分析。

分析閾值

為了減少處理時間並提高整體系統效能，如果自上次執行 ANALYZE 命令後變更的資料列百分比低於 analyze_threshold_percent 參數指定的分析閾值，則 Amazon Redshift 會略過資料表的 ANALYZE 操作。`analyze_threshold_percent` 預設為 10。若要變更目前工作階段的 `analyze_threshold_percent`，請執行 SET 命令。下列範例會將 `analyze_threshold_percent` 變更為 20%。

```
set analyze_threshold_percent to 20;
```

若要在只有少量資料列變更時分析資料表，請將 `analyze_threshold_percent` 設定為任何更小的數。例如，如果將`analyze_threshold_percent` 設為 0.01，則 100,000,000 資料列的資料表中變更的資料列大於 10,000 列資料時，便不會略過資料表。

```
set analyze_threshold_percent to 0.01;
```

若 ANALYZE 因為不符合分析閾值而略過資料表，Amazon Redshift 會傳回以下訊息。

```
ANALYZE SKIP
```

若要分析所有資料表，即使沒有任何資料列變更，請將 analyze_threshold_percent 設為 0。

若要檢視 ANALYZE 操作的結果，請查詢 STL_ANALYZE 系統資料表。

如需分析資料表的相關資訊，請參閱 分析資料表。

範例

分析 TICKIT 資料庫中的所有資料表，並傳回進度資訊。

```
analyze verbose;
```

只分析 LISTING 資料表。

```
analyze listing;
```

分析 VENUE 資料表中的 VENUEID 和 VENUENAME 資料欄。

```
analyze venue(venueid, venuename);
```

只分析 VENUE 資料表中的述詞資料欄。

```
analyze venue predicate columns;
```

ANALYZE COMPRESSION

執行壓縮分析並產生報告，當中針對分析的資料表提供建議的壓縮編碼。對於每一個資料欄，報告都包含與 RAW 編碼相比，可能會減少磁碟空間的估計值。

語法

```
ANALYZE COMPRESSION
[ [ table_name ]
[ ( column_name [, ...] ) ] ]
```

```
[COMPROWS numrows]
```

參數

table_name

您可以分析特定資料表的壓縮，包括臨時資料表。您可以利用結構描述名稱限定資料表。您可以選擇性的指定 table_name 來分析單一資料表。如果您未指定 table_name 值，目前連接的資料庫中所有資料表都會加以分析。您無法在單一 ANALYZE COMPRESSION 陳述式中指定多個 table_name。

column_name

若您指定 table_name，則您也可以在資料表中指定一個或多個資料欄 (括號內以資料欄分隔的清單)。

COMPROWS

資料列數，做為壓縮分析的樣本大小。分析是以每個資料配量中的列為對象。例如，假設您指定 COMPROWS 1000000 (1,000,000)，而系統總共包含 4 個分割，則每個分割最多讀取和分析 250,000 列。如果不指定 COMPROWS，則每個分割的樣本大小預設為 100,000。如果 COMPROWS 的值小於每個分割預設的 100,000 列，則會自動提高到預設值。但是，如果資料表中的資料量不足，無法產生有意義的樣本，則壓縮分析不會產生建議。如果 COMPROWS 數字大於資料表中的資料列數，ANALYZE COMPRESSION 命令仍會繼續，並對所有可用的資料列進行壓縮分析。

numrows

資料列數，做為壓縮分析的樣本大小。numrows 可接受的範圍是介於 1000 到 1000000000 (1,000,000,000) 之間的數字。

使用須知

ANALYZE COMPRESSION 會取得獨佔的資料表鎖定，防止同時對資料表進行讀取和寫入操作。只有在資料表閒置時才執行 ANALYZE COMPRESSION 命令。

執行 ANALYZE COMPRESSION 可根據資料表內容的樣本，取得資料欄編碼機制的建議。ANALYZE COMPRESSION 是建議使用的工具，並且不要修改資料表的資料欄編碼。您可以透過重新建立資料表，或建立具有相同結構描述的新資料表，套用建議的編碼。利用適當的編碼機制重新建立未經壓縮的資料表，可大幅減少在磁碟上佔用的空間。此方法節省了磁碟空間，並提高 I/O 繫結工作負載的查詢效能。

ANALYZE COMPRESSION 會略過實際的分析階段，並直接在指定為 SORTKEY 的任何資料欄上傳回原始編碼類型。這樣做的原因是，當 SORTKEY 資料欄的壓縮程度比其他資料欄高出許多時，限制範圍的掃描執行效果可能較差。

範例

下列範例只顯示 LISTING 資料表中資料欄的編碼和估計減少的百分比：

```
analyze compression listing;

  Table  |     Column     | Encoding | Est_reduction_pct
---------+----------------+----------+-------------------
 listing | listid         | az64     | 40.96
 listing | sellerid       | az64     | 46.92
 listing | eventid        | az64     | 53.37
 listing | dateid         | raw      | 0.00
 listing | numtickets     | az64     | 65.66
 listing | priceperticket | az64     | 72.94
 listing | totalprice     | az64     | 68.05
 listing | listtime       | az64     | 49.74
```

以下範例會分析 SALES 資料表中的 QTYSOLD、COMMISSION 和 SALETIME 資料欄。

```
analyze compression sales(qtysold, commission, saletime);

 Table |   Column   | Encoding | Est_reduction_pct
-------+------------+----------+-------------------
 sales | salesid    | N/A      | 0.00
 sales | listid     | N/A      | 0.00
 sales | sellerid   | N/A      | 0.00
 sales | buyerid    | N/A      | 0.00
 sales | eventid    | N/A      | 0.00
 sales | dateid     | N/A      | 0.00
 sales | qtysold    | az64     | 83.06
 sales | pricepaid  | N/A      | 0.00
 sales | commission | az64     | 71.85
 sales | saletime   | az64     | 49.63
```

ATTACH MASKING POLICY

將現有的動態資料遮罩政策附加至資料欄。如需動態資料遮罩的相關資訊，請參閱 動態資料遮罩。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以附加遮罩政策。

語法

```
ATTACH MASKING POLICY policy_name
   ON { relation_name }
   ( {output_columns_names | output_path} ) [ USING ( {input_column_names | input_path
 )} ]
   TO { user_name | ROLE role_name | PUBLIC }
   [ PRIORITY priority ];
```

參數

policy_name

欲附加的遮罩政策名稱。

relation_name

欲附加遮罩政策的關係名稱。

output_column_names

要套用遮罩政策的資料欄名稱。

output_paths

套用遮罩政策之 SUPER 物件的完整路徑，包括資料欄名稱。例如，對於具有名為 SUPER 類型欄 person 的關係，output_path 可能是 person.name.first_name。

input_column_names

將作為遮罩政策輸入的資料欄名稱。此為選用參數。如果未指定，遮罩政策會使用 output_column_names 做為輸入。

input_paths

遮罩政策會做為輸入之 SUPER 物件的完整路徑。此為選用參數。如果未指定，遮罩政策會使用 output_path 做為輸入。

user_name

要附加遮罩政策的使用者名稱。您無法將兩個政策附加到使用者和欄或角色和欄的相同組合。您可以將一個政策附加至使用者，並將另一個政策附加至使用者的角色。在此情況下，會套用優先順序較高的政策。

您只能在單一 ATTACH MASKING POLICY 命令中設定 user_name、role_name 和 PUBLIC 中的其中一個。

role_name

要附加遮罩政策的角色名稱。您無法將兩個政策附加至相同的資料欄/角色配對。您可以將一個政策附加至使用者，並將另一個政策附加至使用者的角色。在此情況下，會套用優先順序較高的政策。

您只能在單一 ATTACH MASKING POLICY 命令中設定 user_name、role_name 和 PUBLIC 中的其中一個。

PUBLIC

將遮罩政策附加至存取資料表的所有使用者。您必須為附加至特定資料欄/使用者或資料欄/角色配對的其他遮罩政策提供高於 PUBLIC 政策的優先順序，才能套用這些政策。

您只能在單一 ATTACH MASKING POLICY 命令中設定 user_name、role_name 和 PUBLIC 中的其中一個。

priority

遮罩政策的優先順序。當多個遮罩政策套用至指定使用者的查詢時，會套用優先順序最高的政策。

您無法將兩個不同的政策附加到具有相同優先順序的同個資料欄，即使這兩個策略附加到不同的使用者或角色也是如此。只要附加政策的使用者或角色每次都不同，您就可以將相同的政策多次附加至同一組資料表、輸出資料欄、輸入資料欄和優先順序參數。

您無法對資料欄套用與附加至該資料欄的另一個政策具有相同優先順序的政策，即使用於不同角色也一樣。此欄位為選用欄位。如果未指定優先順序，遮罩政策會預設套用值為 0 的優先順序。

ATTACH RLS POLICY

將資料表上的資料列層級安全性政策附加至一或多個使用者或角色。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以附加政策。

語法

```
ATTACH RLS POLICY policy_name ON [TABLE] table_name [, ...]
TO { user_name | ROLE role_name | PUBLIC } [, ...]
```

參數

policy_name

政策的名稱。

ON [TABLE] table_name [, ...]

連接至關係之資料列層級安全政策的名稱。

TO { user_name | ROLE role_name | PUBLIC} [, ...]

指定政策是否附加至一或多個指定的使用者或角色。

使用須知

使用 ATTACH RLS POLICY 陳述式時，請注意以下內容：

- 要附加的資料表應該包含政策建立陳述式的 WITH 子句中列出的所有資料欄。
- Amazon Redshift RLS 不支援將 RLS 政策附加到下列物件：
  - 目錄表
  - 跨資料庫關係
  - 外部資料表
  - 具體化視觀表
  - 暫時資料表
  - 查詢表
- 您無法將 RLS 政策附加至超級使用者，或具有 `sys:secadmin` 權限的使用者。

範例

下列範例會將資料表上的政策附加至角色。

```
ATTACH RLS POLICY policy_concerts ON tickit_category_redshift TO ROLE analyst, ROLE
 dbadmin;
```

BEGIN

開始交易。相當於 START TRANSACTION。

交易是單一邏輯工作單位，無論是由一個命令或多個命令所組成。一般而言，交易中的所有命令會在資料庫的快照上執行，其開始時間是由 transaction_snapshot_begin 系統組態參數的值組所決定。

根據預設，個別 Amazon Redshift 操作 (查詢、DDL 陳述式、負載) 會自動遞交至資料庫。如果您想要暫停遞交某項操作，直到後續工作完成為止，則需使用 BEGIN 陳述式開啟交易，然後執行所需的命令，再使用 COMMIT 或 結束 陳述式關閉交易。若有需要，您可以使用 ROLLBACK 陳述式停止進行中的交易。此行為的例外狀況是 TRUNCATE 命令，它會遞交本身執行所在的交易，而且無法轉返。

語法

```
BEGIN [ WORK | TRANSACTION ] [ ISOLATION LEVEL option ] [ READ WRITE | READ ONLY ]

START TRANSACTION [ ISOLATION LEVEL option ] [ READ WRITE | READ ONLY ]

Where option is

SERIALIZABLE
| READ UNCOMMITTED
| READ COMMITTED
| REPEATABLE READ

Note: READ UNCOMMITTED, READ COMMITTED, and REPEATABLE READ have no
operational impact and map to SERIALIZABLE in Amazon Redshift. You can see database
 isolation levels on your cluster
by querying the stv_db_isolation_level table.
```

參數

WORK

選用的關鍵字。

TRANSACTION

選用的關鍵字；WORK 和 TRANSACTION 為同義詞。

ISOLATION LEVEL SERIALIZABLE

可序列化隔離是預設支援的操作，因此無論陳述式中是否包含此語法，交易的行為都一樣。如需詳細資訊，請參閱 管理並行寫入操作。不支援任何其他隔離層級。

> **Note**
>
> SQL 標準定義了四種交易隔離層級來防止已變更讀取 (交易讀取的資料是由未遞交的同時交易所撰寫)、不可重複讀取 (交易重複讀取先前讀取的資料，且發現該資料已在初次讀取後，經過其他遞交的交易變更)，以及幽靈讀取 (交易再次執行查詢、傳回一組滿足搜尋條件的資料列，然後發現該組資料列已因為其他最近遞交的交易而變更)：
>
> - 讀取未遞交資料：已變更讀取、不可重複讀取及幽靈讀取都有可能發生。
> - 讀取已遞交資料：不可重複讀取及幽靈讀取都有可能發生。
> - 可重複讀取：可能發生幽靈讀取。
> - 可序列化：防止已變更讀取、不可重複讀取及幽靈讀取。
>
> 雖然您可以使用四種交易隔離層級中的任一種，但是 Amazon Redshift 會以可序列化的方式處理所有隔離層級。

READ WRITE

為交易提供讀取和寫入許可。

READ ONLY

為交易提供唯讀許可。

範例

下列範例會啟動可序列化交易區塊：

```
begin;
```

下列範例會啟動具有可序列化隔離層級和讀取與寫入許可的交易區塊：

```
begin read write;
```

CALL

執行預存程序。CALL 命令必須包含程序名稱和輸入引數值。您必須使用 CALL 陳述式來呼叫預存程序。

> **Note**
>
> CALL 不能是任何一般查詢的一部分。

語法

```
CALL sp_name ( [ argument ] [, ...] )
```

參數

sp_name

要執行的程序的名稱。

argument

輸入引數的值。此參數也可以是函數名稱，例如 `pg_last_query_id()`。您不能將查詢當作 CALL 引數。

使用須知

Amazon Redshift 預存程序支援巢狀和遞迴呼叫，如下所述。此外，請確保您的驅動程序支持 up-to-date，還描述如下。

主題

- 巢狀呼叫
- 驅動程式支援

巢狀呼叫

Amazon Redshift 預存程序支援巢狀和遞迴呼叫。允許的巢狀層級數目上限為 16。巢狀呼叫可以將商業邏輯封裝成較小的程序，供多個發起人共用。

如果您呼叫的巢狀程序有輸出參數，則內部程序必須定義 INOUT 引數。在此情況下，非常數變數會傳入內部程序中。不允許 OUT 引數。發生此行為是因為需要變數來保留內部呼叫的輸出。

內層和外層程序的關係記錄在 SVL_STORED_PROC_CALL 的 `from_sp_call` 欄。

下列範例顯示透過 INOUT 引數將變數傳給巢狀程序。

```
CREATE OR REPLACE PROCEDURE inner_proc(INOUT a int, b int, INOUT c int) LANGUAGE
 plpgsql
AS $$
BEGIN
  a := b * a;
  c := b * c;
END;
$$;

CREATE OR REPLACE PROCEDURE outer_proc(multiplier int) LANGUAGE plpgsql
AS $$
DECLARE
  x int := 3;
  y int := 4;
BEGIN
  DROP TABLE IF EXISTS test_tbl;
  CREATE TEMP TABLE test_tbl(a int, b varchar(256));
  CALL inner_proc(x, multiplier, y);
  insert into test_tbl values (x, y::varchar);
END;
$$;

CALL outer_proc(5);

SELECT * from test_tbl;
 a  | b
----+----
 15 | 20
(1 row)
```

驅動程式支援

建議您將 Java 資料庫連線 (JDBC) 和開放式資料庫連線 (ODBC) 驅動程式升級到支援 Amazon Redshift 預存程序的最新版本。

如果您的用戶端工具使用驅動程式 API 操作將 CALL 陳述式傳遞到伺服器，您或許能夠使用現有的驅動程式。傳回的輸出參數 (若有) 是一列的結果集。

最新版本的 Amazon Redshift JDBC 和 ODBC 驅動程式對於預存程序探索，支援中繼資料。對於自訂 Java 應用程式。還支援 `CallableStatement`。如需驅動程式的相關資訊，請參閱《Amazon Redshift 管理指南》中的使用 SQL 用戶端工具連線至 Amazon Redshift 叢集。

下列範例顯示如何在預存程序呼叫中使用 JDBC 驅動程式的不同 API 操作。

```
void statement_example(Connection conn) throws SQLException {
  statement.execute("CALL sp_statement_example(1)");
}

void prepared_statement_example(Connection conn) throws SQLException {
  String sql = "CALL sp_prepared_statement_example(42, 84)";
  PreparedStatement pstmt = conn.prepareStatement(sql);
  pstmt.execute();
}

void callable_statement_example(Connection conn) throws SQLException {
  CallableStatement cstmt = conn.prepareCall("CALL sp_create_out_in(?,?)");
  cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
  cstmt.setInt(2, 42);
  cstmt.executeQuery();
  Integer out_value = cstmt.getInt(1);
}
```

範例

下列範例呼叫程序名稱 test_spl。

```
call test_sp1(3,'book');
INFO:  Table "tmp_tbl" does not exist and will be skipped
INFO:  min_val = 3, f2 = book
```

下列範例呼叫程序名稱 test_spl2。

```
call test_sp2(2,'2019');

         f2          | column2
---------------------+---------
 2019+2019+2019+2019 | 2
(1 row)
```

取消

取消目前正在執行的資料庫查詢。

CANCEL 命令需要執行中查詢的處理序 ID 或工作階段識別碼，並顯示確認訊息以確認查詢是否已取消。

所需權限

以下是 CANCEL 所需的權限：

- 超級使用者取消自己的查詢
- 超級使用者取消使用者的查詢
- 具有 CANCEL 權限的使用者取消使用者的查詢
- 使用者取消自己的查詢

語法

```
CANCEL process_id [ 'message' ]
```

參數

process_id

若要取消在 Amazon Redshift 叢集中執行的查詢，請使用與您要取消STV_RECENTS之查詢對應的 pid (處理 ID)。

若要取消在 Amazon Redshift 無伺服器工作群組中執行的查詢，請使用與您要session_id取SYS_QUERY_HISTORY消之查詢對應的寄件者。

'message'

選用的確認訊息，會在查詢取消完成時顯示。如果您未指定訊息，Amazon Redshift 會顯示預設訊息進行驗證。您必須將訊息用單引號括住。

使用須知

您無法透過指定查詢 ID 來取消查詢；您必須指定查詢的處理序 ID (PID) 或工作階段 ID。您只能取消目前由您的使用者執行的查詢。超級使用者可以取消所有查詢。

如果多個工作階段中的查詢在同一個資料表上保留鎖定，您可以使用 PG_TERMINATE_BACKEND 函數終止其中一個工作階段。這麼做可強制終止的工作階段中任何目前執行中的交易釋出所有鎖定和復原交易。若要檢視目前的鎖定情形，可查詢 STV_LOCKS 系統資料表。

在特定內部事件後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 PID。如果 PID 已變更，您可能會收到下列錯誤訊息。

```
Session <PID> does not exist. The session PID might have changed. Check the
 stl_restarted_sessions system table for details.
```

若要尋找新的 PID，請查詢 STL_RESTARTED_SESSIONS 系統資料表並依 oldpid 資料欄篩選。

```
select oldpid, newpid from stl_restarted_sessions where oldpid = 1234;
```

範例

若要取消 Amazon Redshift 叢集中目前執行中的查詢，請先擷取要取消之查詢的處理序 ID。若要判斷所有目前執行中查詢的處理程序 ID，請輸入下列命令：

```
select pid, starttime, duration,
trim(user_name) as user,
trim (query) as querytxt
from stv_recents
where status = 'Running';

pid |         starttime          | duration |   user   |    querytxt
-----+----------------------------+----------+----------+-----------------
802 | 2008-10-14 09:19:03.550885 |      132 | dwuser | select
venuename from venue where venuestate='FL', where venuecity not in
('Miami' , 'Orlando');
834 | 2008-10-14 08:33:49.473585 |  1250414 | dwuser | select *
from listing;
964 | 2008-10-14 08:30:43.290527 |   326179 | dwuser | select
sellerid from sales where qtysold in (8, 10);
```

查看查詢文字，以判斷哪一個處理程序 ID (PID) 對應到您要取消的查詢。

輸入下列命令，以使用 PID 802 取消該查詢：

```
cancel 802;
```

查詢執行所在的工作階段會顯示以下訊息：

```
ERROR:  Query (168) cancelled on user's request
```

其中 168 是查詢 ID (不是用來取消查詢的處理程序 ID)。

或者，您可以指定顯示自訂確認訊息來取代預設訊息。如要指定自訂訊息，請在 CANCEL 命令的結尾以單引號括住您的訊息：

```
cancel 802 'Long-running query';
```

查詢執行所在的工作階段會顯示以下訊息：

```
ERROR:  Long-running query
```

CLOSE

(選用) 關閉所有與開啟游標相關的可用資源。COMMIT、結束 和 ROLLBACK 會自動關閉游標，因此不需要使用 CLOSE 命令明確關閉游標。

如需詳細資訊，請參閱 DECLARE、FETCH。

語法

```
CLOSE cursor
```

參數

cursor

要關閉的游標名稱。

CLOSE 範例

以下命令會關閉游標並執行遞交，進而結束交易：

```
close movie_cursor;
commit;
```

COMMENT

建立或變更有關資料庫物件的註解。

語法

```
COMMENT ON
```

```
{
TABLE object_name |
COLUMN object_name.column_name |
CONSTRAINT constraint_name ON table_name |
DATABASE object_name |
VIEW object_name
}
IS 'text' | NULL
```

參數

object_name

要加上註解之資料庫物件的名稱。您可以新增註解至下列物件：

- TABLE
- COLUMN (也會取用 column_name)。
- CONSTRAINT (也會取用 constraint_name 和 table_name)。
- DATABASE
- VIEW
- 結構描述

IS 'text' | NULL

您要對指定物件新增或取代的註解文字。文字字串是 TEXT 資料類型。以單引號括住註解。將值設定為 NULL 可移除註解文字。

column_name

要加上註解之資料欄的名稱。COLUMN 的參數。接在 object_name 中指定的資料表後面。

constraint_name

要加上註解之限制條件的名稱。CONSTRAINT 的參數。

table_name

包含限制條件的資料表名稱。CONSTRAINT 的參數。

使用須知

您必須是超級使用者或資料庫物件的擁有者，才能新增或更新註解。

資料庫的註解僅適用於目前資料庫。如果您嘗試對不同資料庫加上註解，則會顯示警告訊息。對不存在的資料庫加上註解時，也會顯示同樣的警告。

不支援對外部表格、外部資料行和後續繫結檢視的資料行的註解。

範例

下列範例會將註解新增至 SALES 資料表。

```
COMMENT ON TABLE sales IS 'This table stores tickets sales data';
```

下列範例會在 SALES 資料表上顯示註解。

```
select obj_description('public.sales'::regclass);

obj_description
-------------------------------------
This table stores tickets sales data
```

下列範例會從 SALES 資料表中移除註解。

```
COMMENT ON TABLE sales IS NULL;
```

下列範例會將註解新增至 SALES 資料表的 EVENTID 資料欄。

```
COMMENT ON COLUMN sales.eventid IS 'Foreign-key reference to the EVENT table.';
```

下列範例會在 SALES 資料表的 EVENTID 資料欄 (資料欄編號 5) 上顯示註解。

```
select col_description( 'public.sales'::regclass, 5::integer );

col_description
----------------------------------------
Foreign-key reference to the EVENT table.
```

下列範例會將描述性註解新增至 EVENT 資料表。

```
comment on table event is 'Contains listings of individual events.';
```

若要檢視註解，請查詢 PG_DESCRIPTION 系統目錄。以下範例會傳回 EVENT 資料表的描述。

```
select * from pg_catalog.pg_description
where objoid =
(select oid from pg_class where relname = 'event'
and relnamespace =
(select oid from pg_catalog.pg_namespace where nspname = 'public') );

objoid | classoid | objsubid | description
-------+----------+----------+---------------------------------------
116658 |     1259 |        0 | Contains listings of individual events.
```

COMMIT

將目前交易遞交至資料庫。此命令會使交易的資料庫更新變成永久有效。

語法

```
COMMIT [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。預存程序內不支援使用此關鍵字。

TRANSACTION

選用的關鍵字。WORK 和 TRANSACTION 為同義詞，預存程序內不支援使用這兩個關鍵字。

如需在預存程序內使用 COMMIT 的相關資訊，請參閱管理交易。

範例

以下每個範例都會將目前交易遞交至資料庫：

```
commit;
```

```
commit work;
```

```
commit transaction;
```

COPY

從資料檔案或 Amazon DynamoDB 資料表，將資料載入資料表。檔案可能位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體、Amazon EMR 叢集，或使用 Secure Shell (SSH) 連線來存取的遠端主機。

> **Note**
>
> Amazon Redshift Spectrum 外部資料表是唯讀的。您無法 COPY 到外部資料表。

COPY 命令會將新的輸入資料附加到資料表中任何現有的列。

任何來源的單一輸入資料列最大為 4 MB。

主題

所需的許可

若要使用 COPY 命令，您必須擁有 Amazon Redshift 資料表的 INSERT 權限。

COPY 語法

```
COPY table-name
```

```
[ column-list ]
FROM data_source
authorization
[ [ FORMAT ] [ AS ] data_format ]
[ parameter [ argument ] [, ... ] ]
```

您可以只使用三個參數來執行 COPY 操作：資料表名稱、資料來源、存取資料的授權。

Amazon Redshift 延伸 COPY 命令的功能，可讓您從多個資料來源載入多種資料格式的資料、控制資料載入的存取權限、管理資料轉換，以及管理載入操作。

以下各節介紹必要的 COPY 命令參數，並依功能將選用參數分組。其中也會描述每個參數，並解釋如何搭配運用各種選項。您可以利用依字母順序排列的參數清單，直接跳到參數描述。

必要參數

COPY 命令需要三個元素：

- Table Name
- Data Source
- Authorization

最簡單的 COPY 命令採用下列格式。

```
COPY table-name
FROM data-source
authorization;
```

下列範例會建立名為 CATDEMO 的資料表，然後從 Amazon S3 中一個名為 `category_pipe.txt` 的資料檔案，將範例資料載入此資料表。

```
create table catdemo(catid smallint, catgroup varchar(10), catname varchar(10), catdesc
 varchar(50));
```

在下列範例中，COPY 命令的資料來源是位於名為 `category_pipe.txt` 之 Amazon S3 儲存貯體的 `tickit` 資料夾中，一個名為 `redshift-downloads` 的資料檔案。COPY 命令獲得授權，可透過 AWS Identity and Access Management (IAM) 角色存取 Amazon S3 儲存貯體。如果叢集現有的 IAM 角色已附加 Amazon S3 的存取權，您可以在下列 COPY 命令中換成此角色的 Amazon Resource Name (ARN)，再執行命令。

```
copy catdemo
from 's3://redshift-downloads/tickit/category_pipe.txt'
iam_role 'arn:aws:iam::<aws-account-id>:role/<role-name>'
region 'us-east-1';
```

如需如何使用 COPY 命令載入範例資料的完整指示，包括從其他 AWS 區域載入資料的指示，請參閱 Amazon Redshift 入門指南中的從 Amazon S3 載入範例資料。

table-name

COPY 命令的目標資料表名稱。此資料表必須已存在於資料庫中。此資料表可以是暫時性或持久性。COPY 命令會將新的輸入資料附加到資料表中任何現有的資料列。

FROM data-source

在目標資料表中載入之來源資料的位置。資訊清單檔案可與一些資料來源一起指定。

最常用的資料儲存庫是 Amazon S3 儲存貯體。您也可以從位於 Amazon EMR 叢集、Amazon EC2 執行個體或遠端主機 (叢集可利用 SSH 連線來存取) 的資料檔案載入，或直接從 DynamoDB 資料表載入。

- 從 Amazon S3 進行 COPY
- 從 Amazon EMR 進行 COPY
- 從遠端主機 COPY (SSH)
- 從 Amazon DynamoDB 進行 COPY

授權

指出叢集用於驗證和授權以存取其他 AWS 資源的方法的子句。COPY 命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、亞馬遜動態 B 和 Amazon EC2。您可以參考叢集附加的 IAM 角色，或將存取金鑰 ID 和私密存取金鑰提供給 IAM 使用者，以提供該授權。

- 授權參數
- 角色類型存取控制
- 金鑰型存取控制

選用的參數

您可以選擇指定 COPY 如何將欄位資料映射至目標資料表中的欄、定義來源資料屬性讓 COPY 命令正確讀取和剖析來源資料，以及管理 COPY 命令在載入過程中執行哪些操作。

- 欄映射選項
- 資料格式參數
- 資料轉換參數
- 資料載入操作

欄映射

根據預設，COPY 會依欄位在資料檔案中出現的同樣順序，將欄位值插入目標資料表的欄。如果預設欄順序不適用，您可以指定欄清單或使用 JSONPath 表達式，將來源資料欄位映射至目標欄。

- Column List
- JSONPaths File

資料格式參數

您可以從固定寬度、字元分隔、逗號分隔值 (CSV) 或 JSON 格式的文字檔案，或從 Avro 檔案載入資料。

根據預設，COPY 命令預期來源資料是位於字元分隔的 UTF-8 文字檔案中。預設分隔符號是縱線字元 (|)。如果來源資料是其他格式，請使用下列參數來指定資料格式。

- FORMAT
- CSV
- DELIMITER
- FIXEDWIDTH
- SHAPEFILE
- AVRO
- JSON
- ENCRYPTED
- BZIP2
- GZIP
- LZOP
- PARQUET
- ORC

- ZSTD

資料轉換參數

COPY 載入資料表時會嘗試隱含地將來源資料中的字串轉換為目標欄的資料類型。如果您需要指定不同於預設行為的轉換，或預設轉換造成錯誤，您可以指定下列參數來管理資料轉換。

- ACCEPTANYDATE
- ACCEPTINVCHARS
- BLANKSASNULL
- DATEFORMAT
- EMPTYASNULL
- ENCODING
- ESCAPE
- EXPLICIT_IDS
- FILLRECORD
- IGNOREBLANKLINES
- IGNOREHEADER
- NULL AS
- REMOVEQUOTES
- ROUNDEC
- TIMEFORMAT
- TRIMBLANKS
- TRUNCATECOLUMNS

資料載入操作

指定下列參數來管理載入操作的預設行為，以進行故障排除或縮短載入時間。

- COMPROWS
- COMPUPDATE
- IGNOREALLERRORS
- MAXERROR

- NOLOAD
- STATUPDATE

COPY 命令的使用注意事項和其他資源

如需如何使用 COPY 命令的相關資訊，請參閱下列主題：

- 使用須知
- 教學課程：從 Amazon S3 載入資料
- 載入資料的 Amazon Redshift 最佳實務
- 使用 COPY 命令載入資料
  - 從 Amazon S3 載入資料
  - 從 Amazon EMR 載入資料
  - 從遠端主機載入資料
  - 從 Amazon DynamoDB 資料表載入資料
- 針對資料載入進行故障診斷

COPY 命令範例

如需顯示如何從不同來源、不同格式和不同 COPY 選項進行 COPY 的更多範例，請參閱COPY 範例。

COPY JOB (預覽)

這是預覽版本中的自動複製 (SQL 複製工作) 的發行前文件。文件和功能會隨時變更。我們建議僅在測試環境中使用此功能，不要在生產環境中使用。公開預覽將於 2024 年 7 月 31 日結束。預覽叢集將在預覽版結束的兩週後自動移除。如需預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

如需有關在檢視中使用此命令的資訊，請參閱持續從 Amazon S3 擷取檔案 (預覽)。

管理將資料載入資料表的 COPY 命令。COPY JOB 命令是 COPY 命令的擴充功能，可自動從 Amazon S3 儲存貯體載入資料。當您建立 COPY 任務時，Amazon Redshift 會偵測何時在指定路徑中建立新的 Amazon S3 檔案，然後自動載入這些檔案，而無需您介入。載入資料時，會使用原始 COPY 命令中使用的相同參數。Amazon Redshift 會追蹤載入的檔案，以確認檔案只載入一次。

> **Note**
>
> 如需 COPY 命令的相關資訊，包括用法、參數和許可，請參閱COPY。

所需的許可

若要執行 COPY JOB 的 COPY 命令，您必須具有要載入資料表的 INSERT 權限。

使用 COPY 命令指定的 IAM 角色必須具有存取要載入之資料的許可。如需詳細資訊，請參閱 COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可。

語法

建立複製任務。COPY 命令的參數會與複製任務一起儲存。

```
COPY copy-command JOB CREATE job-name
[AUTO ON | OFF]
```

變更複製任務的組態。

```
COPY JOB ALTER job-name
[AUTO ON | OFF]
```

執行複製任務。會使用儲存的 COPY 命令參數。

```
COPY JOB RUN job-name
```

列出所有複製任務。

```
COPY JOB LIST
```

顯示複製任務的詳細資訊。

```
COPY JOB SHOW job-name
```

刪除複製任務。

```
COPY JOB DROP job-name
```

參數

copy-command

將資料從 Amazon S3 載入到 Amazon Redshift 的 COPY 命令。該子句包含 COPY 參數，用於定義 Amazon S3 儲存貯體、目標資料表、IAM 角色和載入資料時使用的其他參數。支援 Amazon S3 資料載入的所有 COPY 命令參數，但以下參數除外：

- COPY JOB 不會將預先存在的資料夾內擷取 COPY 命令所指向的檔案。只有在 COPY JOB 建立時間戳記之後建立的檔案才會被擷取。
- 您無法使用 MAXERROR 或 IGNOREALLERRORS 選項來指定 COPY 命令。
- 您不能指定資訊清單檔案。COPY JOB 需要指定的 Amazon S3 位置來監控新建立的檔案。
- 您不能指定具有存取金鑰和私密金鑰等授權類型的 COPY 命令。僅支援使用 IAM_ROLE 參數進行授權的 COPY 命令。如需詳細資訊，請參閱 授權參數。
- COPY JOB 不支援與叢集相關聯的預設 IAM 角色。您必須在 COPY 命令中指定 IAM_ROLE。

如需詳細資訊，請參閱 從 Amazon S3 進行 COPY。

job-name

用來參考 COPY 任務之任務的名稱。

[AUTO ON | OFF]

指出是否將 Amazon S3 資料自動載入 Amazon Redshift 資料表的子句。

- ON 時，Amazon Redshift 會監控新建立檔案的來源 Amazon S3 路徑，如果找到，則會使用任務定義中的 COPY 參數執行 COPY 命令。此為預設值。
- OFF 時，Amazon Redshift 不會自動執行 COPY JOB。

使用須知

COPY 命令的選項直到執行時間才驗證。例如，COPY JOB 開始時，無效的 IAM_ROLE 或 Amazon S3 資料來源會導致執行期錯誤。

如果叢集已暫停，則不會執行 COPY JOBS。

若要查詢載入的 COPY 命令檔案和載入錯誤，請參閱 STL_LOAD_COMMITS、STL_LOAD_ERRORS、STL_LOADERROR_DETAIL。如需詳細資訊，請參閱 驗證資料已正確載入。

範例

下列範例示範建立 COPY JOB，以從 Amazon S3 儲存貯體載入資料。

```
COPY public.target_table
FROM 's3://mybucket-bucket/staging-folder'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyLoadRoleName'
JOB CREATE my_copy_job_name
AUTO ON;
```

COPY 參數參考

COPY 具有許多參數，可以在許多情況下使用。但是，並非每種情況都支援所有參數。例如，若要從 ORC 或 PARQUET 檔案載入，支援的參數數目有限。如需詳細資訊，請參閱 從單欄資料格式 COPY。

主題

- 資料來源
- 授權參數
- 欄映射選項
- 資料格式參數
- 檔案壓縮參數
- 資料轉換參數
- 資料載入操作
- 依字母排序的參數清單

資料來源

您可以從位於 Amazon S3 儲存貯體、Amazon EMR 叢集或遠端主機 (叢集可利用 SSH 連線來存取) 的文字檔案來載入資料。您也可以直接從 DynamoDB 資料表載入資料。

任何來源的單一輸入資料列最大為 4 MB。

若要將資料表中的資料匯出至 Amazon S3 中的一組檔案，請使用 UNLOAD 命令。

主題

- 從 Amazon S3 進行 COPY
- 從 Amazon EMR 進行 COPY

- 從遠端主機 COPY (SSH)
- 從 Amazon DynamoDB 進行 COPY

從 Amazon S3 進行 COPY

若要從位於一或多個 S3 儲存貯體的檔案載入資料，請使用 FROM 子句來指出 COPY 如何尋找 Amazon S3 中的檔案。您可以在 FROM 子句中提供資料檔案的物件路徑，或提供資訊清單檔案 (包含 Amazon S3 物件路徑清單) 的位置。來自 Amazon S3 的 COPY 會使用 HTTPS 連線。確定 S3 IP 範圍已新增至您的允許清單。若要進一步了解所需的 S3 IP 範圍，請參閱網路隔離。

> **Important**
>
> 如果保存資料檔案的 Amazon S3 儲存貯體與叢集不在同一個 AWS 區域，您必須使用REGION參數來指定資料所在的區域。

主題

- 語法
- 範例
- 選用的參數
- 不支援的參數

語法

```
FROM { 's3://objectpath' | 's3://manifest_file' }
authorization
| MANIFEST
| ENCRYPTED
| REGION [AS] 'aws-region'
| optional-parameters
```

範例

下列範例使用物件路徑以從 Amazon S3 載入資料。

```
copy customer
from 's3://mybucket/customer'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

下列範例使用資訊清單檔案以從 Amazon S3 載入資料。

```
copy customer
from 's3://mybucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

參數

FROM

載入的資料來源。如需 Amazon S3 檔案編碼的相關資訊，請參閱資料轉換參數。

's3://copy_from_s3_objectpath'

指定含有資料之 Amazon S3 物件的路徑 Amazon S3 - 例如 `'s3://mybucket/custdata.txt'`。s3://copy_from_s3_objectpath 參數可以參考單一檔案，或一組有相同金鑰前綴的物件或資料夾。例如，`custdata.txt` 這個名稱是參考一些實體檔案的金鑰前綴：`custdata.txt`、`custdata.txt.1`、`custdata.txt.2`、`custdata.txt.bak` 等。金鑰前綴也可以參考一些資料夾。例如，`'s3://mybucket/custfolder'` 參考資料夾 `custfolder`、`custfolder_1`、`custfolder_2` 等。如果金鑰前綴參考多個資料夾，則會載入這些資料夾中的所有檔案。如果金鑰前綴符合檔案又符合資料夾，例如 `custfolder.log`，COPY 也會嘗試載入此檔案。如果金鑰前綴可能導致 COPY 嘗試載入不需要的檔案，請使用資訊清單檔案。如需詳細資訊，請參閱下列 copy_from_s3_manifest_file。

> **Important**
>
> 如果保存資料檔案的 S3 儲存貯體與叢集不在同一個 AWS 區域，您必須使用REGION參數來指定資料所在的區域。

如需詳細資訊，請參閱 從 Amazon S3 載入資料。

's3://copy_from_s3_manifest_file'

指定資訊清單檔案 (列出要載入的資料檔案) 的 Amazon S3 物件金鑰。's3://copy_from_s3_manifest_file' 引數必須明確參考單一檔案 - 例如 `'s3://mybucket/manifest.txt'`。無法參考金鑰前綴。

清單檔案是 JSON 格式的文字檔案，其中列出要從 Amazon S3 載入之每個檔案的 URL。URL 包含檔案的儲存貯體名稱和完整物件路徑。資訊清單中指定的檔案可以位於不同的值區中，但所有值區

都必須與 Amazon Redshift 叢集位於相同的 AWS 區域中。如果某個檔案列出兩次，則該檔案會載入兩次。下列範例顯示資訊清單的 JSON，此資訊清單會載入三個檔案。

```
{
  "entries": [
    {"url":"s3://mybucket-alpha/custdata.1","mandatory":true},
    {"url":"s3://mybucket-alpha/custdata.2","mandatory":true},
    {"url":"s3://mybucket-beta/custdata.1","mandatory":false}
  ]
}
```

雙引號字元是必要的，且必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。資訊清單中的每個項目可以選擇性包含 mandatory 旗標。如果 mandatory 設為 true，COPY 找不到該項目的檔案時會終止；否則，COPY 會繼續。mandatory 的預設值為 false。

載入格式為 ORC 或 Parquet 的資料檔案時，需要 meta 欄位，如下列範例所示。

```
{
   "entries":[
      {
         "url":"s3://mybucket-alpha/orc/2013-10-04-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      },
      {
         "url":"s3://mybucket-beta/orc/2013-10-05-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      }
   ]
}
```

即使指定 ENCRYPTED、GZIP、LZOP、BZIP2 或 ZSTD 選項，也不可對資訊清單檔案進行加密或壓縮。如果找不到指定的資訊清單檔案，或資訊清單檔案的格式不正確，COPY 會傳回錯誤。

如果使用資訊清單檔案，則必須使用 COPY 命令指定 MANIFEST 參數。如果未指定 MANIFEST 參數，COPY 會假設 FROM 指定的檔案是資料檔案。

如需詳細資訊，請參閱 從 Amazon S3 載入資料。

authorization

COPY 命令需要取得授權來存取另一個 AWS 資源中 (包括在 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2 中) 的資料。您可以參考附加到叢集的 AWS Identity and Access Management (IAM) 角色 (以角色為基礎的存取控制) 或提供使用者的存取登入資料 (以金鑰為基礎的存取控制) 來提供該授權。為了提高安全性和彈性，建議使用 IAM 角色型存取控制。如需詳細資訊，請參閱 授權參數。

MANIFEST

指定使用清單檔案來指出要從 Amazon S3 載入的資料檔案。如果使用 MANIFEST 參數，COPY 會從 's3://copy_from_s3_manifest_file' 參考的資訊清單中所列的檔案載入資料。如果找不到資訊清單檔案，或此檔案的格式不正確，COPY 會失敗。如需詳細資訊，請參閱 使用資訊清單指定資料檔案。

ENCRYPTED

此子句指定 Amazon S3 上的輸入檔案是以用戶端加密搭配客戶管理金鑰來加密。如需詳細資訊，請參閱 從 Amazon S3 載入加密的資料檔案。如果輸入檔案是以 Amazon S3 伺服器端加密 (SSE-KMS 或 SSE-S3) 所加密，請勿指定 ENCRYPTED。COPY 會自動讀取伺服器端加密檔案。

如果指定 ENCRYPTED 參數，則還必須指定 MASTER_SYMMETRIC_KEY 參數，或在 CREDENTIALS 字串中包含 **master_symmetric_key** 值。

如果加密檔案是壓縮格式，請增加 GZIP、LZOP、BZIP2 或 ZSTD 參數。

即使指定 ENCRYPTED 選項，也不得對資訊清單檔案和 JSONPaths 檔案進行加密。

MASTER_SYMMETRIC_KEY 'root_key'

用於將 Amazon S3 上的資料檔案加密的根對稱金鑰。如果指定 MASTER_SYMMETRIC_KEY，則還必須指定 ENCRYPTED 參數。MASTER_SYMMETRIC_KEY 不能與 CREDENTIALS 參數一起使用。如需詳細資訊，請參閱 從 Amazon S3 載入加密的資料檔案。

如果加密檔案是壓縮格式，請增加 GZIP、LZOP、BZIP2 或 ZSTD 參數。

REGION [AS] 'aws-region'

指定來源資料所在的 AWS 區域。當含有資料的 AWS 資源所在區域與 Amazon Redshift 叢集的區域不是同一個區域時，從 Amazon S3 儲存貯體或 DynamoDB 資料表的 COPY 需要 REGION。

aws_region 的值必須符合 Amazon Redshift 區域與端點資料表所列的區域。

如果指定 REGION 參數，則所有資源 (包括資訊清單檔案或多個 Amazon S3 儲存貯體) 都必須位於指定的區域。

> **Note**
>
> 跨區域傳輸資料需要為含有資料的 Amazon S3 儲存貯體或 DynamoDB 資料表另外付費。如需有關定價的詳細資訊，請參閱 Amazon S3 定價頁面上的資料從 Amazon S3 傳出到其他 AWS 區域和 Amazon DynamoDB 定價頁面上的資料傳出。

根據預設，COPY 會假設資料與 Amazon Redshift 叢集位於相同區域。

選用的參數

從 Amazon S3 COPY 時，您可以選擇性指定下列參數：

- 欄映射選項
- 資料格式參數
- 資料轉換參數
- 資料載入操作

不支援的參數

從 Amazon S3 COPY 時，您不能使用下列參數：

- SSH
- READRATIO

從 Amazon EMR 進行 COPY

您可以使用 COPY 命令從 Amazon EMR 叢集平行載入資料，而該叢集設定為以固定寬度檔案、字元分隔檔案、CSV 檔案、JSON 格式檔案或 Avro 檔案的形式，將文字檔案寫入叢集的 Hadoop 分散式檔案系統 (HDFS)。

主題

- 語法

- 範例
- 參數
- 支援的參數
- 不支援的參數

語法

```
FROM 'emr://emr_cluster_id/hdfs_filepath'
authorization
[ optional_parameters ]
```

範例

下列範例從 Amazon EMR 叢集載入資料。

```
copy sales
from 'emr://j-SAMPLE2B500FC/myoutput/part-*'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

參數

FROM

載入的資料來源。

'emr://emr_cluster_id/hdfs_file_path'

Amazon EMR 叢集的唯一識別碼，以及 HDFS 檔案路徑 (參考 COPY 命令的資料檔案)。HDFS 資料檔案名稱不能包含萬用字元星號 (*) 和問號 (?)。

> **Note**
>
> Amazon EMR 叢集必須一直運作到 COPY 操作完成為止。如果 COPY 操作完成之前有任何 HDFS 資料檔案經變更或遭刪除，可能會發生非預期的結果，COPY 操作也可能失敗。

您可以在 hdfs_file_path 引數中使用萬用字元星號 (*) 和問號 (?) 來指定要載入的多個檔案。例如，`'emr://j-SAMPLE2B500FC/myoutput/part*'` 表示檔案 `part-0000`、`part-0001`，以

此類推。如果檔案路徑不含萬用字元，則視為字串常值。如果僅指定資料夾名稱，COPY 會嘗試載入該資料夾中的所有檔案。

> **Important**
>
> 如果使用萬用字元或只使用資料夾名稱，請確認不會載入不需要的檔案。例如，某些程序可能將日誌檔案寫入至輸出資料夾。

如需詳細資訊，請參閱 從 Amazon EMR 載入資料。

authorization

COPY 命令需要取得授權來存取另一個 AWS 資源中 (包括在 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2 中) 的資料。您可以參考附加到叢集的 AWS Identity and Access Management (IAM) 角色 (以角色為基礎的存取控制) 或提供使用者的存取登入資料 (以金鑰為基礎的存取控制) 來提供該授權。為了提高安全性和彈性，建議使用 IAM 角色型存取控制。如需詳細資訊，請參閱 授權參數。

支援的參數

從 Amazon EMR COPY 時，您可以選擇性指定下列參數：

- 欄映射選項
- 資料格式參數
- 資料轉換參數
- 資料載入操作

不支援的參數

從 Amazon EMR COPY 時，您不能使用下列參數：

- ENCRYPTED
- MANIFEST
- REGION
- READRATIO
- SSH

從遠端主機 COPY (SSH)

您可以使用 COPY 命令從一或多台遠端主機平行載入資料，例如 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體或其他電腦。COPY 會使用 Secure Shell (SSH) 連接至遠端主機，然後在遠端主機執行命令來產生文字輸出。遠端主機可以是 EC2 Linux 執行個體，或另一台設定為接受 SSH 連線的 Unix 或 Linux 電腦。Amazon Redshift 可以連線到多台主機，而且可以對每台主機開啟多個 SSH 連線。Amazon Redshift 會透過每個連線傳送一個唯一的命令，將文字輸出產生到主機的標準輸出，然後 Amazon Redshift 會像讀取文字檔一樣讀取該輸出。

使用 FROM 子句來指定資訊清單檔案的 Amazon S3 物件金鑰，此資訊清單檔案提供讓 COPY 用來開啟 SSH 連線和執行遠端命令的資訊。

主題

> **Important**
>
> 如果存放資訊清單檔案的 S3 儲存貯體所在區域與叢集的 AWS 區域不是同一個，您必須使用 REGION 參數來指定儲存貯體所在的區域。

語法

```
FROM 's3://'ssh_manifest_file' }
authorization
SSH
| optional-parameters
```

範例

下列範例使用資訊清單檔案，以利用 SSH 從遠端主機載入資料。

```
copy sales
```

```
from 's3://mybucket/ssh_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
ssh;
```

參數

FROM

載入的資料來源。

's3://copy_from_ssh_manifest_file'

COPY 命令可以使用 SSH 來連接至多台主機，也可以對每台主機建立多個 SSH 連線。COPY 會透過每個主機連線執行命令，然後將命令的輸出平行載入資料表。s3://copy_from_ssh_manifest_file 引數會指定資訊清單檔案的 Amazon S3 物件金鑰，此清單檔案會提供讓 COPY 用來開啟 SSH 連線和執行遠端命令的資訊。

s3://copy_from_ssh_manifest_file 引數必須明確參考單一檔案；不能是金鑰字首。如下列範例所示：

```
's3://mybucket/ssh_manifest.txt'
```

資訊清單檔案是 JSON 格式的文字檔案，供 Amazon Redshift 用來連接至主機。資訊清單檔案指定 SSH 主機端點，以及要在主機上執行以將資料傳回給 Amazon Redshift 的命令。您可以選擇在每個項目中包含主機公有金鑰、登入使用者名稱及 mandatory 旗標。下列範例顯示的資訊清單檔案會建立兩個 SSH 連線：

```
{
    "entries": [
     {"endpoint":"<ssh_endpoint_or_IP>",
           "command": "<remote_command>",
           "mandatory":true,
           "publickey": "<public_key>",
           "username": "<host_user_name>"},
     {"endpoint":"<ssh_endpoint_or_IP>",
           "command": "<remote_command>",
           "mandatory":true,
           "publickey": "<public_key>",
           "username": "<host_user_name>"}
     ]
}
```

資訊清單檔案包含一個 "entries" 結構來含括每個 SSH 連線。您可以對單一主機建立多個連線，也可以對多台主機建立多個連線。如圖所示，欄位名稱和值都需要雙引號字元。引號字元必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。唯一不需要雙引號字元的值是 "mandatory" 欄位的布林值 true 或 false。

下列清單描述資訊清單檔案中的欄位。

端點

主機的 URL 地址或 IP 地址，例如 "ec2-111-222-333.compute-1.amazonaws.com" 或 "198.51.100.0"。

command

主機執行的命令，會產生文字輸出或 gzip、lzop、bzip2 或 zstd 格式的二進位輸出。命令可以是使用者 "host_user_name" 有許可執行的任何命令。命令可能像是列印檔案這麼簡單，也可能是查詢資料庫或啟動指令碼。輸出 (文字檔案、gzip 二進位檔案、lzop 二進位檔案，或 bzip2 二進位檔案) 必須是 Amazon Redshift COPY 命令可擷取的格式。如需詳細資訊，請參閱 準備您的輸入資料。

publickey

(選用) 主機的公有金鑰。如果提供，Amazon Redshift 會使用公有金鑰來識別主機。如果未提供公有金鑰，Amazon Redshift 不會嘗試識別主機。例如，若遠端主機的公有金鑰是 ssh-rsa AbcCbaxxx…Example root@amazon.com，請在公有金鑰欄位中輸入下列文字："AbcCbaxxx…Example"

mandatory

(選用) 此子句指出如果嘗試連線失敗，COPY 命令是否就應該失敗。預設值為 false。如果 Amazon Redshift 未成功建立至少一個連線，則 COPY 命令會失敗。

使用者名稱

(選用) 用來登入主機系統並執行遠端命令的使用者名稱。使用者登入名稱與用來將 Amazon Redshift 叢集公有金鑰新增至主機授權金鑰檔案的登入必須相同。預設使用者名稱為 redshift。

如需建立資訊清單檔案的相關資訊，請參閱載入資料程序。

若要從遠端主機進行 COPY，則必須使用 COPY 命令指定 SSH 參數。如果未指定 SSH 參數，COPY 會假設 FROM 指定的檔案是資料檔案，而且將會失敗。

如果您使用自動壓縮，COPY 命令會執行兩次資料讀取操作，也就是會執行遠端命令兩次。第一次讀取操作會提供資料樣本進行壓縮分析，第二次讀取操作就會實際載入資料。如果遠端命

令的兩次執行可能造成問題，請停用自動壓縮。若要停用自動壓縮，請在執行 COPY 命令時將 COMPUPDATE 參數設為 OFF。如需詳細資訊，請參閱 利用自動壓縮載入資料表。

關於使用從 SSH COPY 的詳細程序，請參閱從遠端主機載入資料。

authorization

COPY 命令需要取得授權來存取另一個 AWS 資源中 (包括在 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2 中) 的資料。您可以參考附加到叢集的 AWS Identity and Access Management (IAM) 角色 (以角色為基礎的存取控制) 或提供使用者的存取登入資料 (以金鑰為基礎的存取控制) 來提供該授權。為了提高安全性和彈性，建議使用 IAM 角色型存取控制。如需詳細資訊，請參閱 授權參數。

SSH

此子句指定從使用 SSH 通訊協定的遠端主機載入資料。如果指定 SSH，則還必須使用 s3://copy_from_ssh_manifest_file 引數提供資訊清單檔案。

> **Note**
>
> 如果您使用 SSH 從位在遠端 VPC 使用私有 IP 地址的主機複製，該 VPC 必須已啟用增強型 VPC 路由。如需增強型 VPC 路由的相關資訊，請參閱 Amazon Redshift 增強型 VPC 路由。

選用的參數

從 SSH COPY 時，您可以選擇性指定下列參數：

- 欄映射選項
- 資料格式參數
- 資料轉換參數
- 資料載入操作

不支援的參數

從 SSH COPY 時，您不能使用下列參數：

- ENCRYPTED
- MANIFEST

- READRATIO

從 Amazon DynamoDB 進行 COPY

若要從現有的 DynamoDB 資料表載入資料，請使用 FROM 子句來指定 DynamoDB 資料表名稱。

主題

> **Important**
>
> 如果 DynamoDB 資料表所在區域與 Amazon Redshift 叢集的區域不是同一個，您必須使用 REGION 參數來指定資料所在的區域。

語法

```
FROM 'dynamodb://table-name'
authorization
READRATIO ratio
| REGION [AS] 'aws_region'
| optional-parameters
```

範例

下列範例從 DynamoDB 資料表載入資料。

```
copy favoritemovies from 'dynamodb://ProductCatalog'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
readratio 50;
```

參數

FROM

載入的資料來源。

'dynamodb://table-name'

包含資料之 DynamoDB 資料表的名稱，例如 `'dynamodb://ProductCatalog'`。如需 DynamoDB 屬性如何映射至 Amazon Redshift 欄的詳細資訊，請參閱從 Amazon DynamoDB 資料表載入資料。

DynamoDB 表格名稱對於帳戶而言是唯一的，該 AWS 帳戶由 AWS 存取認證識別。

authorization

COPY 命令需要取得授權來存取另一個 AWS 資源中 (包括在 Amazon S3、Amazon EMR、DynamoDB 和 Amazon EC2 中) 的資料。您可以參考附加到叢集的 AWS Identity and Access Management (IAM) 角色 (以角色為基礎的存取控制) 或提供使用者的存取登入資料 (以金鑰為基礎的存取控制) 來提供該授權。為了提高安全性和彈性，建議使用 IAM 角色型存取控制。如需詳細資訊，請參閱 授權參數。

READRATIO [AS] 比率

DynamoDB 資料表的佈建輸送量中用於資料載入的百分比。從 DynamoDB 的 COPY 需要 READRATIO。此項目不能用於從 Amazon S3 的 COPY。強烈建議將此比率設定為小於平均未用佈建輸送量的值。有效值為整數 1–200。

> **Important**
>
> 將 READRATIO 設為 100 或更高會使 Amazon Redshift 完全耗盡 DynamoDB 資料表的佈建輸送量，導致 COPY 工作階段期間對相同資料表同時執行的讀取操作效能嚴重降低。寫入流量不受影響。在 Amazon Redshift 無法滿足資料表的佈建輸送量的罕見情況下，允許使用高於 100 的值來排解此問題。如果您持續從 DynamoDB 將資料載入 Amazon Redshift，請考慮以時間序列來組織 DynamoDB 資料表，以分隔來自 COPY 操作的即時流量。

選用的參數

從 Amazon DynamoDB COPY 時，您可以選擇性指定下列參數：

- 欄映射選項
- 支援下列資料轉換參數：
  - ACCEPTANYDATE
  - BLANKSASNULL

- DATEFORMAT
- EMPTYASNULL
- ROUNDEC
- TIMEFORMAT
- TRIMBLANKS
- TRUNCATECOLUMNS

- 資料載入操作

不支援的參數

從 DynamoDB COPY 時，您不能使用下列參數：

- 所有資料格式參數
- ESCAPE
- FILLRECORD
- IGNOREBLANKLINES
- IGNOREHEADER
- NULL
- REMOVEQUOTES
- ACCEPTINVCHARS
- MANIFEST
- ENCRYPTED

授權參數

COPY 命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、亞馬遜動態 B 和 Amazon EC2。您可以透過參考附加到叢集的 AWS Identity and Access Management (IAM) 角色 來提供此授權 (角色型存取控制)。您可以在 Amazon S3 上加密載入資料。

下列主題提供身分驗證選項的詳細資訊和範例：

- COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可
- 角色類型存取控制

- 金鑰型存取控制

使用下列其中一項來提供 COPY 命令的授權：

- IAM_ROLE 參數
- ACCESS_KEY_ID and SECRET_ACCESS_KEY 參數
- CREDENTIALS 子句

IAM_ROLE { default | 'arn:aws:iam::*<AWS ##-id>*:role/*<role-name>*' }

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設值並在 COPY 命令執行時與叢集關聯的 IAM 角色。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。如果指定 IAM_ROLE，則不能使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY、SESSION_TOKEN 或 CREDENTIALS。

以下顯示 IAM_ROLE 參數的語法。

```
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
```

如需詳細資訊，請參閱 角色類型存取控制。

存取鍵識別碼 '' 秘密存取金鑰 '秘密存取金鑰 access-key-id '

不建議使用此授權方法。

> **Note**
>
> 除了以純文字提供存取登入資料，強烈建議指定 IAM_ROLE 參數來使用角色型身分驗證。如需詳細資訊，請參閱 角色類型存取控制。

SESSION_TOKEN 'temporary-token'

用於暫時存取登入資料的工作階段字符。指定 SESSION_TOKEN 時，您還必須使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY 來提供暫時存取金鑰登入資料。如果指定 SESSION_TOKEN，則不能使用 IAM_ROLE 或 CREDENTIALS。如需詳細資訊，請參閱《IAM 使用者指南》中的暫時安全憑證。

> **Note**
>
> 除了建立暫時安全登入資料，強烈建議使用角色型身分驗證。使用 IAM 角色來授權時，Amazon Redshift 會自動為每個工作階段建立暫時使用者憑證。如需詳細資訊，請參閱角色類型存取控制。

以下顯示 SESSION_TOKEN 參數及 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY 參數的語法。

```
ACCESS_KEY_ID '<access-key-id>'
SECRET_ACCESS_KEY '<secret-access-key>'
SESSION_TOKEN '<temporary-token>';
```

如果指定 SESSION_TOKEN，則不能使用 CREDENTIALS 或 IAM_ROLE。

[WITH] CREDENTIALS [AS] 'credentials-args'

指示叢集存取其他包含資料檔案或資訊清單檔案的 AWS 資源時，將使用的方法的子句。CREDENTIALS 參數不能與 IAM_ROLE 或 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY 一起使用。

> **Note**
>
> 若要提高彈性，建議使用 IAM_ROLE 參數，而不是 CREDENTIALS 參數。

(選用) 如果使用 ENCRYPTED 參數，則 credentials-args 字串也提供加密金鑰。

credentials-args 字串區分大小寫，且不得包含空格。

關鍵字 WITH 和 AS 是選用的且會被忽略。

您可指定為 role-based access control 或 key-based access control。在任一情況下，IAM 角色或使用者必須具有存取指定之 AWS 資源所需的許可。如需詳細資訊，請參閱 COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可。

> **Note**
>
> 為了保護您的 AWS 認證並保護敏感資料，我們強烈建議您使用角色型存取控制。

若要指定角色型存取控制，請依下列格式提供 credentials-args 字串。

```
'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
```

若要使用暫時字符登入資料，您必須提供暫時存取金鑰 ID、暫時私密存取金鑰及暫時字符。credentials-args 字串採用下列格式。

```
CREDENTIALS
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;token=<temporary-token>'
```

如需詳細資訊，請參閱 暫時安全憑證。

如果使用 ENCRYPTED 參數，則 credentials-args 字串採用下列格式，其中 *<root-key>* 是用於加密檔案之根金鑰的值。

```
CREDENTIALS
'<credentials-args>;master_symmetric_key=<root-key>'
```

例如，下列 COPY 命令會使用角色型存取控制搭配加密金鑰。

```
copy customer from 's3://mybucket/mydata'
credentials
'aws_iam_role=arn:aws:iam::<account-id>:role/<role-name>;master_symmetric_key=<root-key>'
```

下列 COPY 命令顯示角色型存取控制搭配加密金鑰。

```
copy customer from 's3://mybucket/mydata'
credentials
'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>;master_symmetric_key=<root-key>'
```

欄映射選項

根據預設，COPY 會依欄位在資料檔案中出現的同樣順序，將值插入目標資料表的欄。如果預設欄順序不適用，您可以指定欄清單或使用 JSONPath 表達式，將來源資料欄位映射至目標欄。

- Column List

- JSONPaths File

資料欄清單

您可以指定逗號分隔的欄名稱清單，以便將來源資料欄位載入特定的目標欄。COPY 陳述式中的欄可以是任何順序，但從純文字檔 (例如 Amazon S3 儲存貯體) 載入時，欄的順序必須符合來源資料的順序。

從 Amazon DynamoDB 資料表載入時，順序並不重要。COPY 命令會將擷取自 DynamoDB 資料表之項目中的屬性名稱，與 Amazon Redshift 資料表中的欄名稱進行比對。如需更多資訊，請參閱從 Amazon DynamoDB 資料表載入資料

欄清單的格式如下。

```
COPY tablename (column1 [,column2, ...])
```

如果從欄清單中省略目標資料表的某一欄，COPY 會載入目標欄的 DEFAULT 表達式。

如果目標欄沒有預設值，COPY 會嘗試載入 NULL。

如果 COPY 嘗試將 NULL 指派給定義為 NOT NULL 的欄，COPY 命令會失敗。

如果欄清單包含 IDENTITY 欄，則還必須指定 EXPLICIT_IDS；如果 IDENTITY 欄遭省略，則不能指定 EXPLICIT_IDS。如果未指定欄清單，則命令的行為會如同已指定完整、按順序的欄清單一樣，而如果也未指定 EXPLICIT_IDS，則會 IDENTITY 欄會遭到省略。

如果資料欄是使用 GENERATED BY DEFAULT AS IDENTITY 定義的，則可以複製它。值是利用您提供的值來產生或更新。EXPLICIT_IDS 不是必要選項。COPY 不會更新身分高浮水印。如需詳細資訊，請參閱 GENERATED BY DEFAULT AS IDENTITY。

JSONPaths 檔案

從 JSON 或 Avro 格式的資料檔案載入時，COPY 會自動將 JSON 或 Avro 來源資料中的資料元素映射至目標資料表的欄。它會比對 Avro 結構定義中的欄位名稱與目標資料表或欄清單中的欄名稱，以達到這個目標。

在某些情況下，您的欄名稱和欄位名稱不相符，或者您需要映射至資料階層中的更深層級。在這些情況下，您可以使用 JSONPaths 檔案將 JSON 或 Avro 資料元素明確地映射至欄。

如需詳細資訊，請參閱 JSONPaths 檔案。

資料格式參數

根據預設，COPY 命令會預期來源資料是字元分隔的 UTF-8 文字。預設分隔符號是縱線字元 (|)。如果來源資料是其他格式，請使用下列參數來指定資料格式：

- FORMAT
- CSV
- DELIMITER
- FIXEDWIDTH
- SHAPEFILE
- AVRO
- JSON
- PARQUET
- ORC

從 Amazon S3 COPY 時，除了標準資料格式，COPY 還支援下列單欄資料格式：

- ORC
- PARQUET

支援從單欄格式進行 COPY，但有某些限制。如需詳細資訊，請參閱 從單欄資料格式 COPY。

資料格式參數

FORMAT [AS]

(選用) 識別資料格式關鍵字。FORMAT 引數描述如下。

CSV [QUOTE [AS] 'quote_character']

在輸入資料中啟用 CSV 格式。若要自動逸出分隔符號、新行字元及換行字元，請以 QUOTE 參數指定的字元來括住欄位。預設引號字元為雙引號 (")。在欄位內使用引號字元時，請多加一個引號字元來逸出此字元。例如，假設引號字元是雙引號，若要插入 `A "quoted" word` 字串，則輸入檔案應該包含 `"A ""quoted"" word"` 字串。使用 CSV 參數時，預設分隔字元為逗號 (,)。您可以使用 DELIMITER 參數來指定不同的分隔字元。

以引號括住欄位時，分隔字元和引號字元之間的空格會被忽略。如果分隔字元是空格字元 (例如 Tab 字元)，則不會將分隔字元視為空格。

CSV 不能與 FIXEDWIDTH、REMOVEQUOTES 或 ESCAPE 一起使用。

QUOTE [AS] 'quote_character'

選用。指定字元做為使用 CSV 參數時的引號字元。預設值為雙引號 (")。如果使用 QUOTE 參數來定義雙引號以外的引號字元，則不需要在欄位內逸出雙引號。QUOTE 參數只能與 CSV 參數一起使用。AS 關鍵字為選用。

DELIMITER [AS] ['delimiter_char']

指定單一 ASCII 字元，用以分隔輸入檔案中的欄位，例如縱線字元 (|)、逗號 (,) 或 Tab 字元 (\t)。支援非列印 ASCII 字元。ASCII 字元也可以用八進位表示，格式為 '\ddd'，其中 'd' 是八進位數字 (0–7)。預設分隔字元為縱線字元 (|)，除非使用 CSV 參數 (在此情況下，預設分隔字元為逗號 (,))。AS 關鍵字為選用。DELIMITER 不能與 FIXEDWIDTH 一起使用。

FIXEDWIDTH 'fixedwidth_spec'

從每個欄寬是固定長度 (而不是以分隔字元隔開的欄) 的檔案載入資料。fixedwidth_spec 是字串，指定使用者定義的欄標籤和欄寬。欄標籤可以是文字字串或整數 (視使用者的選擇而定)。欄標籤與欄名稱無關。標籤/寬度配對的順序必須完全符合資料表欄的順序。FIXEDWIDTH 不能與 CSV 或 DELIMITER 一起使用。在 Amazon Redshift 中，CHAR 和 VARCHAR 欄的長度以位元組表示，因此在準備要載入的檔案時，請確保您指定的欄寬可容納多位元組字元的二進位長度。如需詳細資訊，請參閱 字元類型。

fixedwidth_spec 的格式如下所示：

```
'colLabel1:colWidth1,colLabel:colWidth2, ...'
```

SHAPEFILE [SIMPLIFY [AUTO] ['tolerance']]

在輸入資料中啟用 SHAPEFILE 格式。依預設，Shapefile 的第一欄是 GEOMETRY 或 IDENTITY 欄。所有後續的欄都遵循 Shapefile 中指定的順序。

您不能使用具有 FIXEDWIDTH、REMOVEQUOTES 或 ESCAPE 的 SHAPEFILE。

若要搭配 GEOGRAPHY 物件使用 COPY FROM SHAPEFILE，請先擷取至 GEOMETRY 欄中，然後將物件轉換為 GEOGRAPHY 物件。

SIMPLIFY [tolerance]

(選用) 使用 Ramer-Douglas-Peucker 演算法和指定的公差，簡化擷取過程中的所有幾何。

SIMPLIFY AUTO [tolerance]

(選用) 僅簡化大於最大幾何大小的幾何。此簡化使用 Ramer-Douglas-Peucker 演算法和自動計算的公差 (如果未超過指定的公差)。此演算法會在指定公差內計算儲存物件的大小。tolerance 值是選用的。

如需載入 Shapefile 的範例，請參閱 將 Shapefile 載入 Amazon Redshift。

AVRO [AS] 'avro_option'

指定來源資料是 Avro 格式。

從這些服務和通訊協定進行 COPY 時，支援 Avro 格式：

- Amazon S3
- Amazon EMR
- 遠端主機 (SSH)

從 DynamoDB 進行 COPY 時不支援 Avro。

Avro 是資料序列化通訊協定。Avro 來源檔案包含結構描述來定義資料的結構。Avro 結構描述類型必須是 `record`。COPY 接受使用預設未壓縮解碼器及 `deflate` 和 `snappy` 壓縮解碼器建立的 Avro 檔案。如需 Avro 的相關資訊，請前往 Apache Avro。

avro_option 的有效值如下所示：

- `'auto'`
- `'auto ignorecase'`
- `'s3://jsonpaths_file'`

預設值為 `'auto'`。

COPY 會自動將 Avro 來源資料中的資料元素映射至目標資料表中的欄。它會比對 Avro 結構描述中的欄位名稱與目標資料表中的欄名稱。`'auto'` 的比對區分大小寫，而 `'auto ignorecase'` 的比對不區分大小寫。

Amazon Redshift 資料表中的欄名稱一律為小寫，因此當您使用 `'auto'` 選項時，相符欄位名稱也必須是小寫。如果欄位名稱並非全部小寫，您可以使用 `'auto ignorecase'` 選項。使用預設 `'auto'` 引數，COPY 只會識別結構中的第一層欄位 (或外部欄位)。

若要將欄名稱明確映射至 Avro 欄位名稱，您可以使用 JSONPaths 檔案。

根據預設，COPY 會嘗試將目標資料表中的所有欄與 Avro 欄位名稱進行比對。若要載入欄子集，您可以選擇性指定欄清單。如果從欄清單中省略目標資料表的某一欄，COPY 會載入目標欄的 DEFAULT 運算式。如果目標欄沒有預設值，COPY 會嘗試載入 NULL。如果某一欄出現在欄清單中，且 COPY 在 Avro 資料中找不到相符欄位，則 COPY 會嘗試將 NULL 載入此欄。

如果 COPY 嘗試將 NULL 指派給定義為 NOT NULL 的欄，COPY 命令會失敗。

Avro 結構描述

Avro 來源資料檔案包含結構描述來定義資料的結構。COPY 會讀取 Avro 來源資料檔案中的結構描述，以便將資料元素映射至目標資料表欄。下列範例顯示 Avro 結構描述。

```
{
    "name": "person",
    "type": "record",
    "fields": [
        {"name": "id", "type": "int"},
        {"name": "guid", "type": "string"},
        {"name": "name", "type": "string"},
        {"name": "address", "type": "string"}]
}
```

Avro 結構描述定義方式是 JSON 格式。最上層 JSON 物件包含三個名稱值對，包含名稱 (或金鑰)、"name"、"type" 和 "fields"。

內含物件陣列的 "fields" 金鑰對會定義資料結構中每個欄位的名稱和資料類型。根據預設，COPY 會自動將欄位名稱與欄名稱進行比對。欄名稱一律為小寫，因此相符的欄位名稱也必須是小寫，除非您指定 'auto ignorecase' 選項。不符合欄名稱的任何欄位名稱會被忽略。順序並不重要。在上述範例中，COPY 映射至欄名稱 id、guid、name 和 address。

使用預設的 'auto' 引數時，COPY 只會將第一層物件與欄進行比對。若要映射至結構描述中更深的層級，或欄位名稱和欄名稱不相符，請使用 JSONPaths 檔案來定義映射。如需詳細資訊，請參閱 JSONPaths 檔案。

如果與金鑰相關聯的值是複合 Avro 資料類型 (例如位元組、陣列、記錄、映射或連結)，COPY 會以字串形式載入值。在這裡，字串是資料的 JSON 表示法。COPY 會以字串形式載入 Avro 列舉資料類型，其中內容是類型的名稱。如需範例，請參閱從 JSON 格式 COPY。

Avro 檔案標頭 (包括結構描述和檔案中繼資料) 的大小上限為 1 MB。

單一 Avro 資料區塊的大小上限為 4 MB。這不同於資料列大小上限。如果超過單一 Avro 資料區塊的大小上限，即使產生的資料列小於 4 MB 資料列大小限制，COPY 命令也會失敗。

在計算列大小時，Amazon Redshift 會在內部將縱線字元 (|) 計算兩次。如果輸入資料包含非常多的縱線字元，即使資料區塊小於 4 MB，資料列大小仍可能超過 4 MB。

JSON [AS] 'json_option'

來源資料是 JSON 格式。

從這些服務和通訊協定 COPY 時，支援 JSON 格式：

- Amazon S3
- 從 Amazon EMR 進行 COPY
- 從 SSH COPY

從 DynamoDB 進行 COPY 時不支援 JSON。

json_option 的有效值如下所示：

- `'auto'`
- `'auto ignorecase'`
- `'s3://jsonpaths_file'`
- `'noshred'`

預設值為 `'auto'`。載入 JSON 文件時，Amazon Redshift 不會將 JSON 結構的屬性分解為多個欄。

根據預設，COPY 會嘗試將目標資料表中的所有欄與 JSON 欄位名稱金鑰配對。若要載入欄子集，您可以選擇性指定欄清單。如果 JSON 欄位名稱金鑰不是全部小寫，您可以使用 `'auto ignorecase'` 選項或 JSONPaths 檔案，明確地將欄名稱映射至 JSON 欄位名稱金鑰。

如果從欄清單中省略目標資料表的某一欄，COPY 會載入目標欄的 DEFAULT 表達式。如果目標欄沒有預設值，COPY 會嘗試載入 NULL。如果某一欄出現在欄清單中，且 COPY 在 JSON 資料中找不到相符欄位，則 COPY 會嘗試將 NULL 載入此欄。

如果 COPY 嘗試將 NULL 指派給定義為 NOT NULL 的欄，COPY 命令會失敗。

COPY 會將 JSON 來源資料中的資料元素映射至目標資料表的欄。它會將來源名稱值對中的物件金鑰 (或名稱) 比對目標資料表中的欄名稱來達成此目標。

請參閱每個 json_option 值的下列詳細資訊：

'auto'

使用此選項，比對會區分大小寫。Amazon Redshift 資料表中的欄名稱一律為小寫，當您使用 `'auto'` 選項時，相符的 JSON 欄位名稱也必須是小寫。

'auto ignorecase'

使用此選項，比對不會區分大小寫。Amazon Redshift 資料表中的欄名稱一律為小寫，因此，當您使用 'auto ignorecase' 選項時，對應的 JSON 欄位名稱可以是小寫、大寫或大小寫混合。

's3://jsonpaths_file'

使用此選項，COPY 會使用指名的 JSONPaths 檔案，將 JSON 來源資料中的資料元素映射至目標資料表的欄。*s3://jsonpaths_file* 引數必須是明確參考單一檔案的 Amazon S3 物件金鑰。例如 's3://mybucket/jsonpaths.txt'。引數不能是索引鍵字首。如需使用 JSONPaths 檔案的相關資訊，請參閱 the section called “JSONPaths 檔案”。

在某些情況下，由 jsonpaths_file 指定的檔案具有與 copy_from_s3_objectpath 為資料檔指定的路徑相同的字首。如果是這樣，COPY 會將 JSONPaths 檔案讀取為資料檔案並傳回錯誤。例如，假設您的資料檔案使用物件路徑 s3://mybucket/my_data.json，而您的 JsonPath 檔案是 s3://mybucket/my_data.jsonpaths。在此情況下，COPY 會嘗試將 my_data.jsonpaths 載入為資料檔案。

'noshred'

使用此選項，在載入 JSON 文件時，Amazon Redshift 不會將 JSON 結構的屬性分解為多個欄。

JSON 資料檔案

JSON 資料檔案包含一組物件或陣列。COPY 會將每個 JSON 物件或陣列載入目標資料表中的每一列。對應到一列的每個物件或陣列必須是獨立的根層級結構；亦即，不能是另一個 JSON 結構的成員。

JSON 物件的開頭和結尾是大括號（{}），且包含一組未排序的名稱值對。每一對名稱和值以冒號分隔，而配對以逗號分隔。根據預設，名稱值對中的物件金鑰 (或名稱) 必須符合資料表中相應欄的名稱。Amazon Redshift 資料表中的欄名稱一律為小寫，因此相符的 JSON 欄位名稱金鑰也必須是小寫。如果欄名稱與 JSON 金鑰不符，請使用 the section called “JSONPaths 檔案” 明確地將欄映射至金鑰。

JSON 物件中的順序並不重要。不符合欄名稱的任何名稱會被忽略。以下顯示簡易 JSON 物件的結構。

```
{
```

```
  "column1": "value1",
  "column2": value2,
  "notacolumn" : "ignore this value"
}
```

JSON 陣列 的開頭和結尾是方括號 ([ ])，且包含一組已排序的值 (以逗號分隔)。如果資料檔案使用陣列，您必須指定 JSONPaths 檔案將值與欄配對。以下顯示簡易 JSON 陣列的結構。

```
["value1", value2]
```

JSON 必須格式正確。例如，不能以逗號或其他任何字元 (空格除外) 來分隔物件或陣列。必須以雙引號字元括住字串。引號字元必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。

單一 JSON 物件或陣列的大小上限 (包括大括號或方括號) 為 4 MB。這不同於資料列大小上限。如果超過單一 JSON 物件或陣列的大小上限，即使產生的資料列小於 4 MB 資料列大小限制，COPY 命令也會失敗。

在計算列大小時，Amazon Redshift 會在內部將縱線字元 (|) 計算兩次。如果輸入資料包含非常多的縱線字元，即使物件小於 4 MB，資料列大小仍可能超過 4 MB。

COPY 會載入 \n 做為新行字元，也會載入 \t 做為 Tab 字元。若要載入反斜線，請加上反斜線 (\\ ) 來逸出。

COPY 會在指定的 JSON 來源中搜尋格式正確、有效的 JSON 物件或陣列。如果 COPY 在找到可用 JSON 結構之前或在有效的 JSON 物件或陣列之間遇到任何非空白字元，COPY 會針對每個例項傳回錯誤。這些錯誤都計入 MAXERROR 錯誤計數內。當錯誤計數等於或超過 MAXERROR 時，COPY 會失敗。

對於每個錯誤，Amazon Redshift 會在 STL_LOAD_ERRORS 系統資料表中記錄一列。LINE_NUMBER 欄會記錄造成錯誤之 JSON 物件的最後一行。

如果指定 IGNOREHEADER，COPY 會在 JSON 資料中忽略指定的行數。IGNOREHEADER 計算時一律計數 JSON 資料中的新行字元。

根據預設，COPY 會將空字串載入為空欄位。如果指定 EMPTYASNULL，COPY 會將 CHAR 和 VARCHAR 欄位的空字串載入為 NULL。一律會將其他資料類型 (例如 INT) 的空字串載入為 NULL。

JSON 不支援下列選項：

- CSV

- DELIMITER
- ESCAPE
- FILLRECORD
- FIXEDWIDTH
- IGNOREBLANKLINES
- NULL AS
- READRATIO
- REMOVEQUOTES

如需詳細資訊，請參閱 從 JSON 格式 COPY。如需 JSON 資料結構的相關資訊，請前往 www.json.org。

JSONPaths 檔案

如果您從 JSON 格式或 Avro 來源資料載入，根據預設，COPY 預設會將來源資料中的第一層資料元素映射至目標資料表的欄。它會將名稱值對中的每個名稱 (或物件金鑰) 比對目標資料表中的欄名稱來達成此目標。

如果欄名稱和物件金鑰不符，或若要映射至資料階層中更深的層級，您可以使用 JSONPaths 檔案，以明確地將 JSON 或 Avro 資料元素映射至欄。JSONPaths 檔案會比對目標資料表或欄清單中的欄順序，以便將 JSON 資料元素映射至欄。

JSONPaths 檔案只能包含單一 JSON 物件 (不是陣列)。JSON 物件是名稱值對。物件金鑰 (名稱值對中的名稱) 必須是 "jsonpaths"。名稱值對中的值是 JSONPath 運算式陣列。每個 JSONPath 表達式都參考 JSON 資料階層或 Avro 結構描述中的單一元素，類似於 XPath 表達式參考 XML 文件中元素的方式。如需詳細資訊，請參閱 JSONPath 表達式。

若要使用 JSONPath 檔案，請將 JSON 或 AVRO 關鍵字加入 COPY 命令。使用下列格式指定 JSONPath 檔案的 S3 儲存貯體名稱和物件路徑。

```
COPY tablename
FROM 'data_source'
CREDENTIALS 'credentials-args'
FORMAT AS { AVRO | JSON } 's3://jsonpaths_file';
```

s3://jsonpaths_file 值必須是明確參考單一檔案的 Amazon S3 物件金鑰，例如 's3://mybucket/jsonpaths.txt'。它不能是索引鍵字首。

在某些情況下，如果您是從 Amazon S3 載入，則 jsonpaths_file 指定的檔案具有與 copy_from_s3_objectpath 為資料檔指定的路徑相同的字首。如果是這樣，COPY 會將 JSONPaths 檔案讀取為資料檔案並傳回錯誤。例如，假設您的資料檔案使用物件路徑 s3://mybucket/my_data.json，而您的 JsonPath 檔案是 s3://mybucket/my_data.jsonpaths。在此情況下，COPY 會嘗試將 my_data.jsonpaths 載入為資料檔案。

如果金鑰名稱是 "jsonpaths" 以外的任何字串，COPY 命令不會傳回錯誤，但會忽略 jsonpaths_file，並改用 'auto' 引數。

如果發生下列任何情形，COPY 命令會失敗：

- JSON 格式不正確。
- 有多個 JSON 物件。
- 物件外面存在空格以外的任何字元。
- 陣列元素是空字串或不是字串。

MAXERROR 不會套用至 JSONPaths 檔案。

即使指定 ENCRYPTED 選項，也不可對 JSONPaths 檔案進行加密。

如需詳細資訊，請參閱 從 JSON 格式 COPY。

JSONPath 表達式

JSONPaths 檔案使用 JSONPath 表達式將資料欄位映射至目標欄。每個 JSONPath 運算式都對應至 Amazon Redshift 目標資料表中的一欄。JSONPath 陣列元素的順序必須符合目標資料表或欄清單 (如果使用欄清單) 中的欄順序。

如圖所示，欄位名稱和值都需要雙引號字元。引號字元必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。

如果在 JSON 資料中找不到 JSONPath 表達式所參考的物件元素，COPY 會嘗試載入 NULL 值。如果參考的物件格式不正確，COPY 會傳回載入錯誤。

如果在 JSON 或 Avro 資料中找不到 JSONPath 表達式所參考的陣列元素，COPY 會失敗並傳回下列錯誤：Invalid JSONPath format: Not an array or index out of range. 請從 JSONPaths 中移除任何不存在於來源資料中的陣列元素，並確認來源資料中的陣列格式正確。

JSONPath 運算式可以使用方括號標記法或點標記法，但您不能混用標記法。下列範例示範使用方括號標記法的 JSONPath 表達式。

```
{
    "jsonpaths": [
        "$['venuename']",
        "$['venuecity']",
        "$['venuestate']",
        "$['venueseats']"
    ]
}
```

下列範例示範使用點標記法的 JSONPath 表達式。

```
{
    "jsonpaths": [
        "$.venuename",
        "$.venuecity",
        "$.venuestate",
        "$.venueseats"
    ]
}
```

在 Amazon Redshift COPY 語法的內容中，JSONPath 運算式必須指定 JSON 或 Avro 階層式資料結構中單一名稱元素的明確路徑。Amazon Redshift 不支援任何可能解析為不明確路徑或多個名稱元素的 JSONPath 元素，例如萬用字元或篩選條件運算式。

如需詳細資訊，請參閱 從 JSON 格式 COPY。

使用 JSONPaths 處理 Avro 資料

下列範例示範多個層級的 Avro 結構描述。

```
{
    "name": "person",
    "type": "record",
    "fields": [
        {"name": "id", "type": "int"},
        {"name": "guid", "type": "string"},
        {"name": "isActive", "type": "boolean"},
        {"name": "age", "type": "int"},
        {"name": "name", "type": "string"},
        {"name": "address", "type": "string"},
        {"name": "latitude", "type": "double"},
        {"name": "longitude", "type": "double"},
```

```
        {
            "name": "tags",
            "type": {
                        "type" : "array",
                        "name" : "inner_tags",
                        "items" : "string"
                    }
        },
        {
            "name": "friends",
            "type": {
                        "type" : "array",
                        "name" : "inner_friends",
                        "items" : {
                                    "name" : "friends_record",
                                    "type" : "record",
                                    "fields" : [
                                                {"name" : "id", "type" : "int"},
                                                {"name" : "name", "type" : "string"}
                                               ]
                                  }
                    }
        },
        {"name": "randomArrayItem", "type": "string"}
    ]
}
```

下列範例顯示使用 AvroPath 運算式來參考先前結構描述的 JSONPaths 檔案。

```
{
    "jsonpaths": [
        "$.id",
        "$.guid",
        "$.address",
        "$.friends[0].id"
    ]
}
```

JSONPaths 範例包含下列元素：

jsonpaths

包含 AvroPath運算式的 JSON 物件名稱。

[...]

方括號括住含有路徑元素的 JSON 陣列。

$

貨幣符號表示 Avro 結構描述中的根元素，即 "fields" 陣列。

"$.id",

AvroPath 運算式的目標。在此例子中，目標是 "fields" 陣列中名稱為 "id" 的元素。表達式以逗號分隔。

"$.friends[0].id"

方括號表示陣列索引。JSONPath 表達式採用以零為基礎的索引，所以此表達式是參考 "friends" 陣列中名稱為 "id" 的第一個元素。

Avro 結構描述語法需要使用內部欄位來定義記錄和陣列資料類型的結構。 AvroPath 表示式會忽略內部欄位。例如，欄位 "friends" 定義名為 "inner_friends" 的陣列，此陣列接著定義名為 "friends_record" 的記錄。引用該字段的 AvroPath 表達式"id"可以忽略額外的字段直接引用目標字段。下列 AvroPath 運算式會參考屬於"friends"陣列的兩個欄位。

```
"$.friends[0].id"
"$.friends[0].name"
```

單欄資料格式參數

從 Amazon S3 COPY 時，除了標準資料格式，COPY 還支援下列單欄資料格式。支援從單欄格式 COPY，但有某些限制。如需詳細資訊，請參閱 從單欄資料格式 COPY。

ORC

從採用「最佳化資料列單欄式 (ORC)」檔案格式的檔案載入資料。

PARQUET

從採用 Parquet 檔案格式的檔案載入資料。

檔案壓縮參數

您可以指定下列參數，以便從壓縮資料檔案載入。

檔案壓縮參數

BZIP2

此值指定一或多個 bzip2 壓縮格式的輸入檔案 (.bz2 檔案)。COPY 操作會讀取每個壓縮檔案，並於載入時將資料解壓縮。

GZIP

此值指定一或多個 gzip 壓縮格式的輸入檔案 (.gz 檔案)。COPY 操作會讀取每個壓縮檔案，並於載入時將資料解壓縮。

LZOP

此值指定一或多個 lzop 壓縮格式的輸入檔案 (.lzo 檔案)。COPY 操作會讀取每個壓縮檔案，並於載入時將資料解壓縮。

> **Note**
>
> COPY 不支援以 lzop --filter 選項壓縮的檔案。

ZSTD

此值指定一或多個壓縮 Zstandard 格式的輸入檔案 (.zst 檔案)。COPY 操作會讀取每個壓縮檔案，並於載入時將資料解壓縮。如需詳細資訊，請參閱 ZSTD。

> **Note**
>
> ZSTD 僅支援從 Amazon S3 使用 COPY。

資料轉換參數

COPY 載入資料表時會嘗試隱含地將來源資料中的字串轉換為目標欄的資料類型。如果您需要指定不同於預設行為的轉換，或預設轉換造成錯誤，您可以指定下列參數來管理資料轉換。如需這些參數語法的相關資訊，請參閱 COPY 語法。

- ACCEPTANYDATE
- ACCEPTINVCHARS
- BLANKSASNULL

- DATEFORMAT
- EMPTYASNULL
- ENCODING
- ESCAPE
- EXPLICIT_IDS
- FILLRECORD
- IGNOREBLANKLINES
- IGNOREHEADER
- NULL AS
- REMOVEQUOTES
- ROUNDEC
- TIMEFORMAT
- TRIMBLANKS
- TRUNCATECOLUMNS

資料轉換參數

ACCEPTANYDATE

允許載入任何日期格式而不會產生錯誤，包括無效格式，例如 00/00/00 00:00:00。此參數僅適用於 TIMESTAMP 和 DATE 欄。ACCEPTANYDATE 一定要與 DATEFORMAT 參數一起使用。如果資料的日期格式不符合 DATEFORMAT 規格，Amazon Redshift 會在該欄位中插入 NULL 值。

ACCEPTINVCHARS [AS] ['replacement_char']

即使資料包含無效 UTF-8 字元，也允許將資料載入 VARCHAR 欄。指定 ACCEPTINVCHARS 時，COPY 會以 replacement_char 指定之字元所組成相同長度的字串，取代每個無效 UTF-8 字元。例如，假設替換字元是 '^'，則會以 '^^^' 取代無效的三位元組字元。

替換字元可以是 NULL 除外的任何 ASCII 字元。預設值為問號 (?)。如需無效 UTF-8 字元的相關資訊，請參閱多位元組字元載入錯誤。

COPY 會傳回包含無效 UTF-8 字元的列數，並針對每個受影響的列，在 STL_REPLACEMENTS 系統資料表中新增一筆項目，而每個節點配量最多 100 列。也會取代其他無效 UTF-8 字元，但不會記錄那些取代事件。

如果不指定 ACCEPTINVCHARS，COPY 只要遇到無效 UTF-8 字元就會傳回錯誤。

ACCEPTINVCHARS 僅適用於 VARCHAR 欄。

BLANKSASNULL

將只包含空格字元的空白欄位載入為 NULL。此選項僅適用於 CHAR 和 VARCHAR 欄。一律會將其他資料類型 (例如 INT) 的空白欄位載入為 NULL。例如，會將包含連續三個空白字元 (沒有其他任何字元) 的字串載入為 NULL。預設行為 (不使用此選項時) 是依原狀載入空白字元。

DATEFORMAT [AS] {'dateformat_string' | 'auto' }

如果不指定 DATEFORMAT，則預設格式為 'YYYY-MM-DD'。例如，另一種有效格式為 'MM-DD-YYYY'。

如果 COPY 命令無法辨識日期或時間值的格式，或者，日期或時間值使用不同的格式，請使用 'auto' 引數來搭配 DATEFORMAT 或 TIMEFORMAT 參數。'auto' 引數可以辨識使用 DATEFORMAT 和 TIMEFORMAT 字串時不支援的多種格式。'auto' 關鍵字區分大小寫。如需詳細資訊，請參閱 對 DATEFORMAT 和 TIMEFORMAT 使用自動辨識。

日期格式可以包含時間資訊 (時、分、秒)，但此資訊會遭到忽略。AS 關鍵字為選用。如需詳細資訊，請參閱 DATEFORMAT 和 TIMEFORMAT 字串。

EMPTYASNULL

指出 Amazon Redshift 應該將空的 CHAR 和 VARCHAR 欄位載入為 NULL。其他資料類型 (例如 INT) 的空欄位一律載入為 NULL。當資料包含連續兩個分隔符號，且分隔符號之間沒有字元時，就形成空欄位。EMPTYASNULL 和 NULL AS '' (空字串) 會引起相同的行為。

ENCODING [AS] file_encoding

指定載入資料的編碼類型。COPY 命令會於載入期間將資料從指定的編碼轉換為 UTF-8。

file_encoding 的有效值如下所示：

- UTF8
- UTF16
- UTF16LE
- UTF16BE

預設值為 UTF8。

來源檔案名稱必須使用 UTF-8 編碼。

即使對載入資料指定不同的編碼，下列檔案仍必須使用 UTF-8 編碼：

- 資訊清單檔案

- JSONPaths 檔案

下列參數隨附的引數字串必須使用 UTF-8：

- FIXEDWIDTH 'fixedwidth_spec'
- ACCEPTINVCHARS 'replacement_char'
- DATEFORMAT 'dateformat_string'
- TIMEFORMAT 'timeformat_string'
- NULL AS 'null_string'

固定寬度資料檔案必須使用 UTF-8 編碼。欄位寬度是根據字元數，而不是位元組數。

所有載入資料必須使用指定的編碼。如果 COPY 遇到不同的編碼，則會略過檔案並傳回錯誤。

如果您指定 UTF16，則資料必須有位元組順序標記 (BOM)。如果知道您的 UTF-16 資料是位元組由小到大 (LE) 或位元組由大到小 (BE)，則不論是否有 BOM，您都可以使用 UTF16LE 或 UTF16BE。

ESCAPE

指定此參數時，輸入資料中的反斜線字元 (\) 就視為逸出字元。反斜線字元後面緊接的字元即使通常做為特殊用途，一樣會載入資料表中成為目前欄值的一部分。例如，當分隔符號字元、引號、內嵌的換行符號字元或逸出字元本身是欄值的正當部分時，您可以使用此參數來逸出這些字元。

如果同時指定 ESCAPE 參數和 REMOVEQUOTES 參數，您可以逸出並保留原本可能移除的引號 (' 或 '')。預設 null 字串 (\N) 不受影響，但也可以在輸入資料中以 \\N 逸出。只要不以 NULL AS 參數指定替代 null 字串，\N 和 \\N 會產生相同的結果。

> **Note**
>
> 控制字元 0x00 (NUL) 不能逸出，應該從輸入資料中移除或轉換。此字元視為記錄結束 (EOR) 標記，用於截斷記錄的剩餘部分。

您不能對 FIXEDWIDTH 載入使用 ESCAPE 參數，也不可指定逸出字元本身；逸出字元一律為反斜線字元。您還必須確定輸入資料包含的逸出字元是在適當的位置。

以下一些範例示範輸入資料，以及指定 ESCAPE 參數時載入的資料結果。第 4 列的結果假設也指定 REMOVEQUOTES 參數。輸入資料由縱線分隔的兩個欄位組成：

```
1|The quick brown fox\[newline]
```

```
jumped over the lazy dog.
2| A\\B\\C
3| A \| B \| C
4| 'A Midsummer Night\'s Dream'
```

載入第 2 欄的資料如下所示：

```
The quick brown fox
jumped over the lazy dog.
A\B\C
A|B|C
A Midsummer Night's Dream
```

> **Note**
>
> 使用者必須負責將逸出字元套用至載入的輸入資料。當您重新載入先前以 ESCAPE 參數來卸載的資料時，此規定就沒有必要。在此情況下，資料一定包含必要的逸出字元。

ESCAPE 參數不會解譯八進位、十六進位、Unicode 或其他逸出序列標記法。例如，假設來源資料包含八進位換行值 (\012)，且您嘗試以 ESCAPE 參數載入此資料，Amazon Redshift 會將值 012 載入資料表，但不會將此值解譯為要逸出的換行字元。

在源自 Microsoft Windows 平台的資料中，若要逸出新行字元，您可能需要使用兩個逸出字元：一個用於換行字元，另一個用於新行字元。或者，您可以在載入檔案之前移除換行字元 (例如，使用 dos2unix 公用程式)。

EXPLICIT_IDS

對於有 IDENTITY 欄的資料表，如果您想要以資料表的來源資料檔案中的明確值來覆寫自動產生的值，請使用 EXPLICIT_IDS。如果命令包含欄清單，則此清單必須包含 IDENTITY 欄，才能使用此參數。EXPLICIT_IDS 值的資料格式必須符合 CREATE TABLE 定義所指定的 IDENTITY 格式。

在您以 EXPLICIT_IDS 選項對資料表執行 COPY 命令之後，Amazon Redshift 將不再檢查資料表中的 IDENTITY 欄的唯一性。

如果資料欄是使用 GENERATED BY DEFAULT AS IDENTITY 定義的，則可以複製它。值是利用您提供的值來產生或更新。EXPLICIT_IDS 不是必要選項。COPY 不會更新身分高浮水印。

如需使用 EXPLICIT_IDS 的 COPY 命令範例，請參閱載入 VENUE 時將明確值提供給 IDENTITY 欄。

FILLRECORD

當某些記錄的結尾缺少連續的欄時，允許載入資料檔案。遺失的欄會載入為 NULL。對於文字和 CSV 格式，如果遺失的欄是 VARCHAR 欄，則會載入長度為零的字串，而非 NULL。若要將 NULL 從文字和 CSV 載入至 VARCHAR 欄，請指定 EMPTYASNULL 關鍵字。只有在欄定義允許 NULL 時，才會進行 NULL 替換。

例如，假設資料表定義包含四個可為 Null 的 CHAR 欄，且有一筆記錄包含值 `apple, orange, banana, mango`，則 COPY 命令可以載入並填入只包含值 `apple, orange` 的記錄。會將缺少的 CHAR 值載入為 NULL 值。

IGNOREBLANKLINES

忽略資料檔案中只含有換行字元的空白行，而不嘗試載入這些空白行。

IGNOREHEADER [AS] number_rows

將指定的 number_rows 視為檔案標頭而不載入。使用 IGNOREHEADER 來略過平行載入之所有檔案中的檔案標頭。

NULL AS 'null_string'

將符合 null_string 的欄位載入為 NULL，其中 null_string 可以是任何字串。如果資料包含 null 結束字元 (也稱為 NUL (UTF-8 0000) 或二進位零 (0x000))，COPY 會將其視為任何其他字元。例如，包含 '1' || NUL || '2' 的記錄被複製為長度為 3 個位元組的字串。如果欄位只包含 NUL，您可以使用 NULL AS 並指定 `'\0'` 或 `'\000'`，而以 NULL 取代 null 結束字元 - 例如 `NULL AS '\0'` 或 `NULL AS '\000'`。如果欄位包含以 NUL 結尾的字串，且指定 NULL AS，則會在字串結尾插入 NUL。請勿在 null_string 值使用 '\n' (換行符號)。Amazon Redshift 保留 '\ n' 用作行分隔符號。預設 null_string 為 `'\N'`。

> **Note**
>
> 如果嘗試將 null 載入已定義為 NOT NULL 的欄，COPY 命令會失敗。

REMOVEQUOTES

在傳入的資料中移除括住字串的引號。引號內的所有字元 (包括分隔符號) 都會保留。如果字串有起始單引號或雙引號，但沒有對應的結束標記，則 COPY 命令無法載入此列，且會傳回錯誤。下表以一些簡單的範例示範含有引號的字串及結果載入的值。

輸入字串	以 REMOVEQUOTES 選項載入的值
"分隔符號是縱線 (|) 字元"	分隔符號是縱線 (|) 字元
'黑色'	黑色
"白色"	白色
藍色'	藍色'
藍色'	不會載入值：錯誤情況
"藍色	不會載入值：錯誤情況
'''黑色'''	''黑色''
''	<white space>

ROUNDEC

當輸入值的小數位數超過欄的小數位數時，將數值四捨五入。根據預設，COPY 會視需要截斷值，以符合欄的小數位數。例如，假設將值 20.259 載入 DECIMAL(8,2) 欄，COPY 預設會將值截斷為 20.25。如果指定 ROUNDEC，則 COPY 會將值四捨五入為 20.26。INSERT 命令一律會視需要將值四捨五入，以符合欄的小數位數，因此，COPY 命令搭配 ROUNDEC 參數的行為就如同 INSERT 命令。

TIMEFORMAT [AS] {'timeformat_string' | 'auto' | 'epochsecs' | 'epochmillisecs' }

指定時間格式。如果不指定 TIMEFORMAT，則 TIMESTAMP 欄的預設格式為 YYYY-MM-DD HH:MI:SS，TIMESTAMPTZ 欄的預設格式為 YYYY-MM-DD HH:MI:SSOF，其中 OF 是國際標準時間 (UTC) 的時差。timeformat_string 中不可包含時區指標。若要載入不是預設格式的 TIMESTAMPTZ 資料，請指定 'auto'；如需詳細資訊，請參閱對 DATEFORMAT 和 TIMEFORMAT 使用自動辨識。如需 timeformat_string 的相關資訊，請參閱 DATEFORMAT 和 TIMEFORMAT 字串。

'auto' 引數可以辨識使用 DATEFORMAT 和 TIMEFORMAT 字串時不支援的多種格式。如果 COPY 命令無法辨識日期或時間值的格式，或者，日期和時間值使用彼此不同的格式，請使用 'auto' 引數來搭配 DATEFORMAT 或 TIMEFORMAT 參數。如需詳細資訊，請參閱 對 DATEFORMAT 和 TIMEFORMAT 使用自動辨識。

如果來源資料以 ecpch 時間表示，亦即，自 1970 年 1 月 1 日 00:00:00 UTC 以來的秒數或毫秒數，請指定 'epochsecs' 或 'epochmillisecs'。

'auto'、'epochsecs' 和 'epochmillisecs' 關鍵字區分大小寫。

AS 關鍵字為選用。

TRIMBLANKS

從 VARCHAR 字串中移除尾端空格字元。此參數僅適用於 VARCHAR 資料類型的欄。

TRUNCATECOLUMNS

將欄的資料截斷為適當的字元數，以符合欄規格。僅適用於 VARCHAR 或 CHAR 資料類型的欄，以及大小為 4 MB 或更小的列。

資料載入操作

指定下列參數來管理載入操作的預設行為，以進行故障排除或縮短載入時間。

- COMPROWS
- COMPUPDATE
- IGNOREALLERRORS
- MAXERROR
- NOLOAD
- STATUPDATE

參數

COMPROWS numrows

指定列數做為壓縮分析的樣本大小。分析是以每個資料配量中的列為對象。例如，假設您指定 COMPROWS 1000000 (1,000,000)，而系統總共包含四個配量，則每個配量最多讀取和分析 250,000 列。

如果不指定 COMPROWS，則每個配量的樣本大小預設為 100,000。如果 COMPROWS 的值小於每個配量預設的 100,000 列，則會將該值自動提高到預設值。但是，如果載入的資料量不足以構成有意義的樣本，則不會執行自動壓縮。

如果 COMPROWS 數字大於輸入檔案中的列數，COPY 命令仍會繼續，並對所有可用的列執行壓縮分析。此引數可接受的範圍是介於 1000 到 2147483647 (2,147,483,647) 之間的數字。

COMPUPDATE [PRESET | { ON | TRUE } | { OFF | FALSE }],

控制在 COPY 期間是否自動套用壓縮編碼。

COMPUPDATE 為 PRESET 時，如果目標資料表為空白，即使資料欄已有 RAW 以外的編碼，COPY 命令也會為每個資料欄選擇壓縮編碼。可以取代目前指定的資料欄編碼。每個資料欄的編碼是以資料欄的資料類型為基礎。不會對任何資料取樣。Amazon Redshift 會自動指派壓縮編碼，如下所示：

- 定義為排序索引鍵的資料欄會有指派的 RAW 壓縮。
- 定義為 BOOLEAN、REAL 或 DOUBLE PRECISION 資料類型的資料欄會有指派的 RAW 壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIMESTAMP 或 TIMESTAMPTZ 的資料欄會有指派的 AZ64 壓縮。
- 定義為 CHAR 或 VARCHAR 的資料欄會有指派的 LZO 壓縮。

省略 COMPUPDATE 時，只有在目標資料表為空白而您尚未為任何資料欄指定編碼 (RAW 除外) 時，COPY 命令才會為每個資料欄選擇壓縮編碼。每個欄的編碼是由 Amazon Redshift 決定。不會對任何資料取樣。

當 COMPUPDATE 為 ON (或 TRUE) 或指定了 COMPUPDATE 但未帶選項時，如果資料表是空的，即使資料表欄已有 RAW 以外的編碼，COPY 也會套用自動壓縮。可以取代目前指定的資料欄編碼。每個資料欄的編碼會根據取樣資料的分析。如需詳細資訊，請參閱 利用自動壓縮載入資料表。

將 COMPUPDATE 設為 OFF (或 FALSE) 時，會停用自動壓縮。不會變更資料欄編碼。

如需關於分析壓縮的系統資料表資訊，請參閱 STL_ANALYZE_COMPRESSION。

IGNOREALLERRORS

您可以指定此選項來忽略載入作業期間發生的所有錯誤。

如果您指定 MAXERROR 選項，就無法指定 IGNOREALLERRORS 選項。您無法為包括 ORC 和 Parquet 在內的單欄格式指定 IGNOREALLERRORS 選項。

MAXERROR [AS] error_count

如果載入傳回 error_count 個或更多錯誤，載入會失敗。如果載入傳回的錯誤很少，則會繼續載入，並傳回 INFO 訊息來指出無法載入的列數。當某些列因為格式錯誤或資料有其他不一致情形而無法載入到資料表時，使用此參數可讓載入繼續進行。

如果要讓載入在發生第一個錯誤時立即失敗，請將此值設為 0 或 1。AS 關鍵字為選用。MAXERROR 預設值為 0，限制為 100000。

由於 Amazon Redshift 的平行本質，實際報告的錯誤數可能大於指定的 MAXERROR。如果 Amazon Redshift 叢集中的任何節點偵測到已超過 MAXERROR，每個節點會報告所有已遇到的錯誤。

NOLOAD

檢查資料檔案的有效性，而不實際載入資料。在執行實際資料載入之前，請使用 NOLOAD 參數來確保資料檔案可載入無誤。搭配 NOLOAD 參數來執行 COPY 的速度比載入資料快很多，因為其只會剖析檔案。

STATUPDATE [{ ON | TRUE } | { OFF | FALSE }]

控管在成功的 COPY 命令結束時自動運算和重新整理最佳化工具統計資料。根據預設，如果不使用 STATUPDATE 參數，只要資料表最初是空白，就會自動更新統計資料。

每當將資料擷取到非空白資料表就會大幅改變資料表大小時，建議您執行 ANALYZE 命令或使用 STATUPDATE ON 引數來更新統計資料。

指定 STATUPDATE ON (或 TRUE) 時，不論資料表最初是否空白，都會自動更新統計資料。如果使用 STATUPDATE，則目前使用者必須是資料表擁有者或超級使用者。如果不指定 STATUPDATE，則只需要 INSERT 許可。

指定 STATUPDATE OFF (或 FALSE) 時永遠不會更新統計資料。

如需其他資訊，請參閱 分析資料表。

依字母排序的參數清單

下列清單提供每個 COPY 命令參數描述的連結 (依字母順序排列)。

- ACCEPTANYDATE
- ACCEPTINVCHARS
- ACCESS_KEY_ID and SECRET_ACCESS_KEY
- AVRO
- BLANKSASNULL
- BZIP2
- COMPROWS
- COMPUPDATE

- CREDENTIALS
- CSV
- DATEFORMAT
- DELIMITER
- EMPTYASNULL
- ENCODING
- ENCRYPTED
- ESCAPE
- EXPLICIT_IDS
- FILLRECORD
- FIXEDWIDTH
- FORMAT
- FROM
- GZIP
- IAM_ROLE
- IGNOREALLERRORS
- IGNOREBLANKLINES
- IGNOREHEADER
- JSON
- LZOP
- MANIFEST
- MASTER_SYMMETRIC_KEY
- MAXERROR
- NOLOAD
- NULL AS
- READRATIO
- REGION
- REMOVEQUOTES
- ROUNDEC

- SESSION_TOKEN
- SHAPEFILE
- SSH
- STATUPDATE
- TIMEFORMAT
- SESSION_TOKEN
- TRIMBLANKS
- TRUNCATECOLUMNS
- ZSTD

使用須知

主題

- 存取其他 AWS 資源的許可
- 將 COPY 與 Amazon S3 存取點別名搭配使用
- 從 Amazon S3 載入多位元組資料
- 載入 GEOMETRY 或 GEOGRAPHY 資料類型的欄
- 載入 HLLSKETCH 資料類型
- 載入 VARBYTE 資料類型的欄
- 讀取多個檔案時發生錯誤
- 從 JSON 格式 COPY
- 從單欄資料格式 COPY
- DATEFORMAT 和 TIMEFORMAT 字串
- 對 DATEFORMAT 和 TIMEFORMAT 使用自動辨識

存取其他 AWS 資源的許可

若要在叢集和其他 AWS 資源 (例如 Amazon S3、Amazon DynamoDB、亞馬遜 EMR 或 Amazon EC2) 之間移動資料，您的叢集必須具有存取資源和執行必要動作的權限。例如，若要從 Amazon S3 載入資料，COPY 必須具備儲存貯體的 LIST 存取權和儲存貯體物件的 GET 存取權。如需最低許可的相關資訊，請參閱 COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可。

叢集必須經過驗證，才能獲得授權來存取資源。您可以選擇下列任一種身分驗證方法：

- 角色類型存取控制— 對於以角色為基礎的存取控制，您可以指定叢集用於驗證和授權的 AWS Identity and Access Management (IAM) 角色。為了保護您的 AWS 認證和敏感資料，我們強烈建議您使用以角色為基礎的驗證。
- 金鑰型存取控制— 對於以金鑰為基礎的存取控制，您可以以純文字形式提供使用者的存取認證 (存取金鑰 ID 和秘密存取金鑰)。 AWS

角色類型存取控制

使用角色型存取控制時，叢集會暫時代表您擔任 IAM 角色。然後，根據角色獲予的授權，叢集即可存取所需的 AWS 資源。

建立 IAM 角色類似於授予許可給使用者，因為它是 AWS 身分，具有判定其在 AWS中可執行和不可執行的操作之許可政策。但是，角色可以由任何需要此角色的實體來擔任，而不必與一個使用者產生獨特的關聯。此外，角色沒有任何相關聯的登入資料 (密碼或存取金鑰)。反之，如果角色與叢集相關聯，則會動態建立存取金鑰並將該金鑰提供給叢集。

我們建議您使用以角色為基礎的存取控制，因為除了保護您的認證之外，它還提供對 AWS 資源和敏感使用者資料的存取更加安全、更精細的控制。 AWS

角色型身分驗證提供下列優點：

- 您可以使用 AWS 標準 IAM 工具定義 IAM 角色，並將該角色與多個叢集建立關聯。修改角色的存取政策時，會將變更自動套用至所有使用該角色的叢集。
- 您可以定義精細的 IAM 政策，授予特定叢集和資料庫使用者存取特定 AWS 資源和動作的許可。
- 您的叢集會在執行時間取得臨時工作階段登入資料，並視需要重新整理登入資料，直到操作完成。如果您使用金鑰型臨時登入資料，萬一臨時登入資料在操作完成之前就過期，則操作會失敗。
- 不會在 SQL 程式碼中儲存或傳輸存取金鑰 ID 和私密存取金鑰。

若要使用角色型存取控制，您必須先使用 Amazon Redshift 服務角色類型建立 IAM 角色，再將該角色連接至叢集。角色至少必須具備 COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可中列出的許可。如需建立 IAM 角色並將其附加到叢集的步驟，請參閱 Amazon Redshift 管理指南中的授權 Amazon Redshift 代您存取其他 AWS 服務。

您可以使用 Amazon Redshift 管理主控台、CLI 或 API 將角色新增至叢集，或檢視與叢集相關聯的角色。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的將 IAM 角色與叢集建立關聯。

建立 IAM 角色時，IAM 會傳回角色的 Amazon Resource Name (ARN)。若要指定 IAM 角色，請使用 IAM_ROLE 參數或 CREDENTIALS 參數來提供角色 ARN。

例如，假設下列角色已附加至叢集。

```
"IamRoleArn": "arn:aws:iam::0123456789012:role/MyRedshiftRole"
```

下列 COPY 命令範例使用 IAM_ROLE 參數，搭配上一個範例中的 ARN，以驗證身分並存取 Amazon S3。

```
copy customer from 's3://mybucket/mydata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

下列 COPY 命令範例使用 CREDENTIALS 參數來指定 IAM 角色。

```
copy customer from 's3://mybucket/mydata'
credentials
'aws_iam_role=arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

此外，超級使用者可以將 ASSUMEROLE 權限授與資料庫使用者和群組，以提供 COPY 操作角色的存取權。如需相關資訊，請參閱GRANT。

金鑰型存取控制

透過以金鑰為基礎的存取控制，您可以為獲授權存取包含資料之 AWS 資源的 IAM 使用者提供存取金鑰 ID 和秘密存取金鑰。您可以將 ACCESS_KEY_ID and SECRET_ACCESS_KEY 參數一起使用，或使用 CREDENTIALS 參數。

> **Note**
>
> 強烈建議使用 IAM 角色來驗證身分，而不要提供純文字存取金鑰 ID 和私密存取金鑰。如果您選擇以金鑰為基礎的存取控制，請勿使用您的 AWS 帳號 (root) 憑證。請一律建立 IAM 使用者，並提供該使用者的存取金鑰 ID 和私密存取金鑰。關於建立 IAM 使用者的步驟，請參閱在您的 AWS 帳戶中建立 IAM 使用者。

若要使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY 來驗證身分，請以授權使用者的存取金鑰 ID 和完整私密存取金鑰，取代 *<access-key-id>* 和 *<secret-access-key>*，如下所示。

```
ACCESS_KEY_ID '<access-key-id>'
```

```
SECRET_ACCESS_KEY '<secret-access-key>';
```

若要使用 CREDENTIALS 參數來驗證身分，請以授權使用者的存取金鑰 ID 和完整私密存取金鑰，取代 *<access-key-id>* 和 *<secret-access-key>*，如下所示。

```
CREDENTIALS
'aws_access_key_id=<access-key-id>;aws_secret_access_key=<secret-access-key>';
```

IAM 使用者至少必須具備COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可中列出的許可。

暫時安全憑證

如果使用金鑰型存取控制，您可以利用臨時安全性登入資料，以進一步限制使用者有何權限存取您的資料。角色型身分驗證會自動使用臨時登入資料。

> **Note**
>
> 強烈建議使用role-based access control，而不要建立臨時登入資料並以純文字提供存取金鑰 ID 和私密存取金鑰。角色型存取控制會自動使用暫時憑證。

暫時安全憑證提供加強的安全性，因為有效期限較短，且過期之後不能重複使用。與字符一起產生的存取金鑰 ID 和私密存取金鑰一定要有字符才能使用，而具有這些暫時安全憑證的使用者只能在憑證到期之前存取您的資源。

若要授予使用者暫時存取您的資源，請呼叫 AWS Security Token Service (AWS STS) API 作業。AWS STS API 操作返回由安全令牌，訪問密鑰 ID 和秘密訪問密鑰組成的臨時安全憑據。如果某使用者需要暫時存取您的資源，您可以將臨時安全性登入資料發行給該使用者。這些使用者可以是現有的 IAM 使用者，或非AWS 使用者。如需建立暫時安全憑證的相關資訊，請參閱《IAM 使用者指南》中的使用暫時安全憑證。

您可以將 ACCESS_KEY_ID and SECRET_ACCESS_KEY 參數與 SESSION_TOKEN 參數或 CREDENTIALS 參數一起使用。您也必須提供字符隨附的存取金鑰 ID 和私密存取金鑰。

若要使用 ACCESS_KEY_ID、SECRET_ACCESS_KEY 和 SESSION_TOKEN 來驗證身分，請取代 *<temporary-access-key-id>*、*<temporary-secret-access-key>* 和 *<temporary-token>*，如下所示。

```
ACCESS_KEY_ID '<temporary-access-key-id>'
SECRET_ACCESS_KEY '<temporary-secret-access-key>'
```

```
SESSION_TOKEN '<temporary-token>';
```

若要使用 CREDENTIALS 來驗證身分，請在登入資料字串中包含 session_token=*<temporary-token>*，如下所示。

```
CREDENTIALS
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;session_token=<temporary-token>';
```

下列範例顯示使用臨時安全性登入資料的 COPY 命令。

```
copy table-name
from 's3://objectpath'
access_key_id '<temporary-access-key-id>'
secret_access_key '<temporary-secret-access-key>'
session_token '<temporary-token>';
```

下列範例使用臨時登入資料和檔案加密來載入 LISTING 資料表。

```
copy listing
from 's3://mybucket/data/listings_pipe.txt'
access_key_id '<temporary-access-key-id>'
secret_access_key '<temporary-secret-access-key>'
session_token '<temporary-token>'
master_symmetric_key '<root-key>'
encrypted;
```

下列範例使用 CREDENTIALS 參數搭配臨時登入資料和檔案加密來載入 LISTING 資料表。

```
copy listing
from 's3://mybucket/data/listings_pipe.txt'
credentials
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;session_token=<temporary-token>;master_symmetric_key=<root-key>'
encrypted;
```

> **Important**
>
> 臨時安全性登入資料在 COPY 或 UNLOAD 操作的整個期間內必須有效。如果臨時安全性登入資料在操作期間過期，則命令會失敗，而交易會回復。例如，假設臨時安全性登入資料在 15

分鐘之後過期，而 COPY 操作需要 1 小時，則 COPY 操作在完成之前就會失敗。如果您使用角色型存取，臨時安全性登入資料會自動重新整理，直到操作完成。

COPY、UNLOAD 和 CREATE LIBRARY 的 IAM 許可

CREDENTIALS 參數所參考的 IAM 角色或使用者至少必須具備下列許可：

- 從 Amazon S3 COPY 時，具備許可來 LIST Amazon S3 儲存貯體及 GET 要載入的 Amazon S3 物件和資訊清單檔案 (如果使用)。
- 從 Amazon S3、Amazon EMR 和遠端主機 (SSH) COPY JSON 格式的資料時，具備許可來 LIST 和 GET Amazon S3 上的 JSONPaths 檔案 (如果使用)。
- 從 DynamoDB COPY 時，具備許可來 SCAN 和 DESCRIBE 要載入的 DynamoDB 資料表。
- 從 Amazon EMR 叢集 COPY 時，具備許可在 Amazon EMR 叢集上執行 `ListInstances` 動作。
- UNLOAD 到 Amazon S3 時，對於 Amazon S3 儲存貯體 (要將資料檔案卸載到此處)，具備 GET、LIST 和 PUT 許可。
- 從 Amazon S3 CREATE LIBRARY 時，具備許可來 LIST Amazon S3 儲存貯體及 GET 要匯入的 Amazon S3 物件。

> **Note**
>
> 執行 COPY、UNLOAD 或 CREATE LIBRARY 命令時，如果出現錯誤訊息 `S3ServiceException: Access Denied`，表示叢集沒有 Amazon S3 的適當存取許可。

您可以將 IAM 政策附加至 IAM 角色，而該角色附加至叢集、使用者或使用者所屬的群組，以此來管理 IAM 許可。例如，`AmazonS3ReadOnlyAccess` 受管政策授予 Amazon S3 資源的 LIST 和 GET 許可。如需管理政策的相關資訊，請參閱《IAM 使用者指南》中的管理 IAM 政策。

將 COPY 與 Amazon S3 存取點別名搭配使用

COPY 支援 Amazon S3 存取點別名。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的為您的存取點使用儲存貯體型別名。

從 Amazon S3 載入多位元組資料

如果資料包含非 ASCII 多位元組字元 (例如中文或斯拉夫文字元)，您必須將資料載入 VARCHAR 欄。VARCHAR 資料類型支援四位元組 UTF-8 字元，但 CHAR 資料類型只接受單位元組 ASCII 字

元。您無法將五位元組或更長的字元載入 Amazon Redshift 資料表。如需詳細資訊，請參閱 多位元組字元。

載入 GEOMETRY 或 GEOGRAPHY 資料類型的欄

您可以將以字元分隔的文字檔案 (例如 CSV 檔案) 中的資料複製到 GEOMETRY 或 GEOGRAPHY 欄。資料的格式必須是已知二進位格式 (WKB 或 EWKB) 或已知文字格式 (WKT 或 EWKT) 的十六進位形式，並且符合 COPY 命令的單一輸入列的大小上限。如需詳細資訊，請參閱 COPY。

如需有關如何從 Shapefile 中載入的資訊，請參閱將 Shapefile 載入 Amazon Redshift。

如需 GEOMETRY 或 GEOGRAPHY 資料類型的相關資訊，請參閱在 Amazon Redshift 中查詢空間資料。

載入 HLLSKETCH 資料類型

您只能以 Amazon Redshift 支援的稀疏或密集格式複製 HLL 草圖。若要在 HyperLogLog 草圖上使用 COPY 命令，請針對密集草圖使用 Base64 格式，對稀疏 HyperLogLog HyperLogLog 草圖使用 JSON 格式。如需詳細資訊，請參閱 HyperLogLog 函數。

下列範例使用 CREATE TABLE 和 COPY，將資料從 CSV 檔案匯入資料表。首先，這個範例使用 CREATE TABLE 建立資料表 t1。

```
CREATE TABLE t1 (sketch hllsketch, a bigint);
```

然後它使用 COPY 將資料從 CSV 檔案匯入資料表 t1。

```
COPY t1 FROM s3://DOC-EXAMPLE-BUCKET/unload/' IAM_ROLE
 'arn:aws:iam::0123456789012:role/MyRedshiftRole' NULL AS 'null' CSV;
```

載入 VARBYTE 資料類型的欄

您可以從 CSV、Parquet 和 ORC 格式的檔案載入資料。對於 CSV，資料會以十六進位表示的 VARBYTE 資料從檔案載入。您無法使用 FIXEDWIDTH 選項載入 VARBYTE 資料。不支援 COPY 的 ADDQUOTES 或 REMOVEQUOTES 選項。VARBYTE 欄不能用作分割欄。

讀取多個檔案時發生錯誤

COPY 命令是不可分割和交易式。換言之，即使 COPY 命令從多個檔案讀取資料，整個程序仍視為單一交易。如果 COPY 在讀取檔案時發生錯誤，則會自動重試，直到程序逾時為止 (請參閱 statement_timeout)，或如果經過很久 (15 到 30 分鐘) 仍無法從 Amazon S3 下載資料，請確定每個檔案只載入一次。如果 COPY 命令失敗，整個交易會取消，而所有變更都會回復。如需如何處理載入錯誤的相關資訊，請參閱針對資料載入進行故障診斷。

COPY 命令成功起始之後，即使工作階段終止 (例如用戶端中斷連線)，命令也不會失敗。不過，如果 COPY 命令在 BEGIN … END 交易區塊內，而交易區塊因為工作階段終止而未完成，則整個交易 (包括 COPY) 會回復。如需交易的相關資訊，請參閱 BEGIN。

從 JSON 格式 COPY

JSON 資料結構由一組物件或陣列組成。JSON 物件的開頭和結尾是大括號，且包含一組未排序的名稱值對。每一個名稱和值以冒號分隔，而配對以逗號分隔。名稱是以雙引號括住的字串。引號字元必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。

JSON 陣列 的開頭和結尾是方括號，且包含一組已排序的值 (以逗號分隔)。值可以是以雙引號括住的字串、數字、布林值 true 或 false、null、JSON 物件或陣列。

JSON 物件和陣列可以是巢狀，以支援階層資料結構。下列範例顯示具有兩個有效物件的 JSON 資料結構。

```
{
    "id": 1006410,
    "title": "Amazon Redshift Database Developer Guide"
}
{
    "id": 100540,
    "name": "Amazon Simple Storage Service User Guide"
}
```

下面以兩個 JSON 陣列顯示同樣的資料。

```
[
    1006410,
    "Amazon Redshift Database Developer Guide"
]
[
    100540,
    "Amazon Simple Storage Service User Guide"
]
```

適用於 JSON 的 COPY 選項

將 COPY 與 JSON 格式資料搭配使用時，可以指定下列選項：

- `'auto'` - COPY 會自動從 JSON 檔案載入欄位。

- 'auto ignorecase' - COPY 會自動從 JSON 檔案載入欄位，同時忽略欄位名稱的大小寫。
- s3://jsonpaths_file - COPY 會使用 JSONPaths 檔案來剖析 JSON 來源資料。JSONPaths 檔案是包含單一 JSON 物件的文字檔案，此物件的名稱為 "jsonpaths"，並與 JSONPath 表達式陣列搭配成對。如果名稱是 "jsonpaths" 以外的任何字串，COPY 會使用 'auto' 引數，而不使用 JSONPaths 檔案。

關於如何使用 'auto'、'auto ignorecase' 或 JSONPaths 檔案及使用 JSON 物件或陣列來載入資料的範例，請參閱從 JSON 複製的範例。

JSONPath 選項

在 Amazon Redshift COPY 語法中，JSONPath 運算式會使用括號或點符號，指定 JSON 階層式資料結構中單一名稱元素的明確路徑。Amazon Redshift 不支援任何可能解析為不明確路徑或多個名稱元素的 JSONPath 元素，例如萬用字元或篩選條件運算式。因此，Amazon Redshift 無法剖析複雜、多層級的資料結構。

以下是 JSONPaths 檔案的範例，其中的 JSONPath 表達式使用方括號標記法。貨幣符號 ($) 代表根層級結構。

```
{
    "jsonpaths": [
       "$['id']",
       "$['store']['book']['title']",
 "$['location'][0]"
    ]
}
```

在上述範例中，$['location'][0] 參考陣列的第一個元素。JSON 採用以零為基礎的陣列索引。陣列索引必須是正整數 (大於或等於零)。

下列範例使用點標記法來示範前一個 JSONPaths 檔案。

```
{
    "jsonpaths": [
       "$.id",
       "$.store.book.title",
 "$.location[0]"
    ]
}
```

jsonpaths 陣列中不能混用方括號標記法和點標記法。方括號標記法和點標記法中都可以使用方括號來參考陣列元素。

使用點標記法時，JSONPath 運算式不可包含下列字元：

- 一般單引號 (')
- 句點，或點 (.)
- 方括號 ([])，除非用來參考陣列元素

如果 JSONPath 運算式所參考的名稱值對中的值是物件或陣列，則會將整個物件或陣列載入為字串，包括大括號或方括號。例如，假設 JSON 資料包含下列物件。

```
{
    "id": 0,
    "guid": "84512477-fa49-456b-b407-581d0d851c3c",
    "isActive": true,
    "tags": [
        "nisi",
        "culpa",
        "ad",
        "amet",
        "voluptate",
        "reprehenderit",
        "veniam"
    ],
    "friends": [
        {
            "id": 0,
            "name": "Martha Rivera"
        },
        {
            "id": 1,
            "name": "Renaldo"
        }
    ]
}
```

JSONPath 表達式 $['tags'] 會傳回下列值。

```
"["nisi","culpa","ad","amet","voluptate","reprehenderit","veniam"]"
```

JSONPath 表達式 `$['friends'][1]` 會傳回下列值。

```
"{"id": 1,"name": "Renaldo"}"
```

`jsonpaths` 陣列中的每個 JSONPath 運算式都對應至 Amazon Redshift 目標資料表中的一欄。`jsonpaths` 陣列元素的順序必須符合目標資料表或欄清單 (如果使用欄清單) 中的欄順序。

關於如何使用 `'auto'` 引數或 JSONPaths 檔案及使用 JSON 物件或陣列來載入資料的範例，請參閱從 JSON 複製的範例。

如需如何複製多個 JSON 檔案的相關資訊，請參閱使用資訊清單指定資料檔案。

JSON 中的逸出字元

COPY 會載入 \n 做為新行字元，也會載入 \t 做為 Tab 字元。若要載入反斜線，請加上反斜線 (\\ ) 來逸出。

例如，假設您在 `escape.json` 儲存貯體的一個名為 `s3://mybucket/json/` 檔案中有下列 JSON。

```
{
  "backslash": "This is a backslash: \\",
  "newline": "This sentence\n is on two lines.",
  "tab": "This sentence \t contains a tab."
}
```

執行下列命令來建立 ESCAPES 資料表並載入 JSON。

```
create table escapes (backslash varchar(25), newline varchar(35), tab varchar(35));

copy escapes from 's3://mybucket/json/escape.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as json 'auto';
```

查詢 ESCAPES 資料表來檢視結果。

```
select * from escapes;

       backslash        |      newline      |               tab
------------------------+-------------------+----------------------------------
 This is a backslash: \ | This sentence     | This sentence    contains a tab.
```

```
                        :  is on two lines.
(1 row)
```

遺失數值精確度

從 JSON 格式的資料檔案中將數字載入已定義為數值資料類型的欄時，可能會遺失精確度。電腦系統中無法精確表示某些浮點值。因此，從 JSON 檔案複製的資料可能不會如預期般四捨五入。為了避免遺失精確度，建議您使用下列其中一個替代方法：

- 以字串來表示數字，用雙引號括住值。
- 使用 ROUNDEC 將數字四捨五入而非截斷。
- 使用 CSV、字元分隔或固定寬度的文字檔案，而不要使用 JSON 或 Avro 檔案。

從單欄資料格式 COPY

COPY 可以從 Amazon S3 載入下列單欄格式的資料：

- ORC
- Parquet

如需從單欄式資料格式使用 COPY 的範例，請參閱COPY 範例。

COPY 支援具有下列考量的欄式格式化資料：

- Amazon S3 儲存貯體必須與亞馬遜紅移資料庫位於相同的 AWS 區域。
- 若要透過 VPC 端點存取您的 Amazon S3 資料，請按照《Amazon Redshift 管理指南》中的使用 Amazon Redshift Spectrum 搭配增強型 VPC 路由所述，使用 IAM 政策和 IAM 角色來設定存取。
- COPY 不會自動套用壓縮編碼。
- 僅支援下列 COPY 參數：
  - ACCEPTINVCHARS：從 ORC 或 Parquet 檔案複製時。
  - FILLRECORD
  - FROM
  - IAM_ROLE
  - CREDENTIALS
  - STATUPDATE
  - MANIFEST

  - EXPLICIT_IDS
- 如果 COPY 於載入時發生錯誤，命令會失敗。單欄資料類型不支援 ACCEPTANYDATE 和 MAXERROR。
- 錯誤訊息會傳送給 SQL 用戶端。一些錯誤會記錄在 STL_LOAD_ERRORS 和 STL_ERROR 中。
- COPY 會依欄在單欄資料檔案中出現的同樣順序，將值插入目標資料表的欄。目標資料表的欄數和資料檔案的欄數必須相符。
- 如果您在 COPY 操作中指定的檔案包含下列其中一個副檔名，則不需要新增任何參數，我們就會將資料解壓縮：
  - `.gz`
  - `.snappy`
  - `.bz2`
- 從 Parquet 和 ORC 檔案格式 COPY 需用到 Redshift Spectrum 和儲存貯體存取權。若要針對這些格式使用 COPY，請確定沒有任何 IAM 政策會封鎖 Amazon S3 預先簽署的 URL 的使用。Amazon Redshift 產生的預先簽署網址有效期為 1 小時，因此 Amazon Redshift 有足夠的時間從 Amazon S3 儲存貯體載入所有檔案。COPY 從單欄資料格式掃描的每個檔案都會產生唯一的預先簽署 URL。對於包含`s3:signatureAge`動作的值區政策，請務必將值設定為至少 3,600,000 毫秒。如需詳細資訊，請參閱使用 Amazon Redshift Spectrum 搭配增強型 VPC 路由。

DATEFORMAT 和 TIMEFORMAT 字串

COPY 命令使用 DATEFORMAT 和 TIMEFORMAT 選項來剖析來源資料中的日期和時間值。DATEFORMAT 和 TIMEFORMAT 是格式化字串，必須符合來源資料的日期和時間值的格式。例如，使用日期值 `Jan-01-1999` 載入來源資料的 COPY 命令必須包含下列 DATEFORMAT 字串：

```
COPY ...
            DATEFORMAT AS 'MON-DD-YYYY'
```

如需管理 COPY 資料轉換的相關資訊，請參閱資料轉換參數。

DATEFORMAT 和 TIMEFORMAT 字串可以包含日期時間分隔符號 (例如 '-'、'/' 或 ':')，以及下表中的日期部分和時間部分格式。

> **Note**
>
> 如果您的日期或時間值的格式無法符合以下日期部分和時間部分，或者您的日期和時間值使用的格式彼此不同，請使用 `'auto'` 引數來搭配 DATEFORMAT 或 TIMEFORMAT 參

數。'auto' 引數可以辨識使用 DATEFORMAT 或 TIMEFORMAT 字串時不支援的多種格式。如需詳細資訊，請參閱 對 DATEFORMAT 和 TIMEFORMAT 使用自動辨識。

日期部分或時間部分	意義
YY	年，不含世紀
YYYY	年，包含世紀
MM	月，以數字表示
MON	月，以名稱表示 (縮寫名稱或完整名稱)
DD	月中的日，以數字表示
HH 或 HH24	小時 (24 小時制) **Note** 在 SQL 函數的 DATETIME 格式字串中，HH 與 HH12 相同。不過，在 COPY 的 DATEFORMAT 和 TIMEFORMAT 字串中，HH 與 HH24 相同。
HH12	小時 (12 小時制)
MI	分鐘
SS	秒鐘
AM 或 PM	正午指標 (用於 12 小時制)

預設日期格式為 YYYY-MM-DD。不含時區的預設時間戳記 (TIMESTAMP) 格式為 YYYY-MM-DD HH:MI:SS。具有時區 (TIMESTAMPTZ) 格式的預設時間戳記為 YYYY-MM-DD HH:MI:SSOF，其中 OF 是與 UTC 的偏移量 (例如 -8:00。不能在 timeformat_string 中包含時區指標 (TZ、tz 或 OF)。秒

(SS) 欄位也支援小數秒到微秒的詳細程度。若要載入非預設格式的 TIMESTAMPTZ 資料，請指定 'auto'。

以下是可能在來源資料中遇到的一些範例日期或時間，以及對應的 DATEFORMAT 或 TIMEFORMAT 字串。

來源資料日期或時間範例	DATEFORMAT 或 TIMEFORMAT 語法
03/31/2003	DATEFORMAT AS 'MM/DD/YYYY'
2003 年 3 月 31 日	DATEFORMAT AS 'MON DD, YYYY'
03.31.2003 18:45:05 03.31.2003 18:45:05.123456	TIMEFORMAT AS 'MM.DD.YYYY HH:MI:SS'

範例

如需使用 TIMEFORMAT 的範例，請參閱載入時間戳記或日期戳記。

對 DATEFORMAT 和 TIMEFORMAT 使用自動辨識

如果指定 'auto' 做為 DATEFORMAT 或 TIMEFORMAT 參數的引數，Amazon Redshift 會自動辨識和轉換來源資料中的日期格式或時間格式。下列顯示一個範例。

```
copy favoritemovies from 'dynamodb://ProductCatalog'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
dateformat 'auto';
```

搭配 DATEFORMAT 和 TIMEFORMAT 的 'auto' 引數一起使用時，COPY 會辨識和轉換 DATEFORMAT 和 TIMEFORMAT 字串 中資料表所列出的日期和時間格式。此外，'auto' 引數還可以辨識下列格式 (使用 DATEFORMAT 和 TIMEFORMAT 字串時不支援這些格式)。

格式	有效輸入字串的範例
ISO 8601	2019-02-11T05:09:12.195Z

格式	有效輸入字串的範例
凱撒曆	J2451187
BC	Jan-08-95 BC
YYYYMMDD HHMISS	19960108 040809
YYMMDD HHMISS	960108 040809
YYYY.DDD	1996.008
YYYY-MM-DD HH:MI:SS.SSS	1996-01-08 04:05:06.789
DD Mon HH:MI:SS YYYY TZ	17 Dec 07:37:16 1997 PST
MM/DD/YYYY HH:MI:SS.SS TZ	12/17/1997 07:37:16.00 PST
YYYY-MM-DD HH:MI:SS+/-TZ	1997-12-17 07:37:16-08
DD.MM.YYYY HH:MI:SS TZ	12.17.1997 07:37:16.00 PST

自動辨識不支援 epoch 秒和 epoch 毫秒。

若要測試日期或時間戳記值是否會自動轉換，請使用 CAST 函數來嘗試將字串轉換為日期或時間戳記值。例如，下列命令測試時間戳記值 `'J2345678 04:05:06.789'`：

```
create table formattest (test char(21));
insert into formattest values('J2345678 04:05:06.789');
select test, cast(test as timestamp) as timestamp, cast(test as date) as date from
 formattest;

        test          |      timestamp      | date
----------------------+---------------------+------------
J2345678 04:05:06.789   1710-02-23 04:05:06 1710-02-23
```

如果 DATE 欄的來源資料包含時間資訊，則會截斷時間部分。如果 TIMESTAMP 欄的來源資料省略時間資訊，則時間部分會使用 00:00:00。

COPY 範例

> **Note**
>
> 這些範例包含換行以方便閱讀。請勿在 credentials-args 字串中包含換行或空格。

主題

從 DynamoDB 資料表載入 FAVORITEMOVIES

AWS 開發套件包括建立稱為影片的 DynamoDB 資料表的簡單範例。(關於此範例，請參閱 DynamoDB 入門。) 下列範例將 DynamoDB 資料表中的資料載入 Amazon Redshift MOVIES 資料表。Amazon Redshift 資料表必須已存在於資料庫中。

```
copy favoritemovies from 'dynamodb://Movies'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
readratio 50;
```

從 Amazon S3 儲存貯體載入 LISTING

以下範例從 Amazon S3 儲存貯體載入 LISTING。COPY 命令會載入 /data/listing/ 資料夾中的所有檔案。

```
copy listing
from 's3://mybucket/data/listing/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

從 Amazon EMR 叢集載入 LISTING

下列範例從 Amazon EMR 叢集的 lzop 壓縮檔案中，將 Tab 字元分隔資料載入 SALES 資料表。COPY 會載入 myoutput/ 資料夾中開頭為 part- 的每個檔案。

```
copy sales
from 'emr://j-SAMPLE2B500FC/myoutput/part-*'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '\t' lzop;
```

下列範例將 Amazon EMR 叢集上的 JSON 格式資料載入 SALES 資料表。COPY 會載入 myoutput/json/ 資料夾中的每個檔案。

```
copy sales
from 'emr://j-SAMPLE2B500FC/myoutput/json/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
JSON 's3://mybucket/jsonpaths.txt';
```

使用資訊清單指定資料檔案

您可以使用清單檔案來確保 COPY 命令從 Amazon S3 載入所有必要檔案 (且只有必要檔案)。需要從不同儲存貯體載入多個檔案，或載入不共用相同字首的檔案時，您也可以使用資訊清單。

例如，假設您需要載入下列三個檔案：custdata1.txt、custdata2.txt 和 custdata3.txt。您可以使用下列命令指定字首，以載入 mybucket 中開頭為 custdata 的所有檔案：

```
copy category
from 's3://mybucket/custdata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

如果因為錯誤而只有兩個檔案存在，則 COPY 只會載入那兩個檔案，然後就順利完成，導致資料載入不完整。如果儲存貯體中有一個不需要的檔案剛好也使用相同的字首，例如名為 custdata.backup 的檔案，則 COPY 也會載入此檔案，導致載入不需要的資料。

若要確保載入所有必要檔案並防止載入不需要的檔案，您可以使用資訊清單檔案。資訊清單是 JSON 格式的文字檔案，其中列出要由 COPY 命令處理的檔案。例如，下列資訊清單會載入上述範例中的三個檔案。

```
{
   "entries":[
      {
         "url":"s3://mybucket/custdata.1",
         "mandatory":true
      },
      {
         "url":"s3://mybucket/custdata.2",
         "mandatory":true
      },
      {
         "url":"s3://mybucket/custdata.3",
         "mandatory":true
      }
   ]
}
```

選用的 mandatory 旗標指出如果檔案不存在，則 COPY 是否應該終止。預設值為 false。不考慮任何必要設定，只要找不到檔案，COPY 就會終止。在此範例中，如果找不到任何檔案，COPY 會傳回錯誤。如果您僅指定金鑰前綴 (例如 custdata.backup)，則可能已挑選的不必要檔案會被忽略，因為這些檔案不在資訊清單上。

載入格式為 ORC 或 Parquet 的資料檔案時，需要 meta 欄位，如下列範例所示。

```
{
   "entries":[
```

```
      {
         "url":"s3://mybucket-alpha/orc/2013-10-04-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      },
      {
         "url":"s3://mybucket-beta/orc/2013-10-05-custdata",
         "mandatory":true,
         "meta":{
            "content_length":99
         }
      }
   ]
}
```

以下範例使用名為 cust.manifest 的資訊清單。

```
copy customer
from 's3://mybucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as orc
manifest;
```

您可以使用資訊清單從不同儲存貯體載入檔案，或載入不共用相同字首的檔案。下列範例示範的 JSON 會從名稱開頭為日期戳記的檔案載入資料。

```
{
  "entries": [
    {"url":"s3://mybucket/2013-10-04-custdata.txt","mandatory":true},
    {"url":"s3://mybucket/2013-10-05-custdata.txt","mandatory":true},
    {"url":"s3://mybucket/2013-10-06-custdata.txt","mandatory":true},
    {"url":"s3://mybucket/2013-10-07-custdata.txt","mandatory":true}
  ]
}
```

只要值區與叢集位於相同區域，資訊清單就可以列出位於不同值 AWS 區中的檔案。

```
{
  "entries": [
    {"url":"s3://mybucket-alpha/custdata1.txt","mandatory":false},
```

```
    {"url":"s3://mybucket-beta/custdata1.txt","mandatory":false},
    {"url":"s3://mybucket-beta/custdata2.txt","mandatory":false}
  ]
}
```

從縱線分隔檔案 (預設分隔符號) 載入 LISTING

下列範例是非常簡單的案例，其中沒有指定任何選項，且輸入檔案包含預設分隔符號，即縱線字元 ('|')。

```
copy listing
from 's3://mybucket/data/listings_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

使用 Parquet 格式的單欄資料載入 LISTING

下列範例從 Amazon S3 上名為 parquet 的資料夾載入資料。

```
copy listing
from 's3://mybucket/data/listings/parquet/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as parquet;
```

使用 ORC 格式的單欄資料載入 LISTING

下列範例從 Amazon S3 上名為 orc 的資料夾載入資料。

```
copy listing
from 's3://mybucket/data/listings/orc/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as orc;
```

使用選項載入 EVENT

下列範例將縱線分隔資料載入 EVENT 資料表，並套用下列規則：

- 如果使用成對的引號來括住任何字元字串，則會移除引號。
- 都會將空字串和含有空白的字串載入為 NULL 值。
- 如果傳回 5 個以上的錯誤，載入會失敗。

- 時間戳記值必須符合指定的格式；例如，有效時間戳記為 2008-09-26 05:43:12。

```
copy event
from 's3://mybucket/data/allevents_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
removequotes
emptyasnull
blanksasnull
maxerror 5
delimiter '|'
timeformat 'YYYY-MM-DD HH:MI:SS';
```

從固定寬度資料檔案載入 VENUE

```
copy venue
from 's3://mybucket/data/venue_fw.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth 'venueid:3,venuename:25,venuecity:12,venuestate:2,venueseats:6';
```

上述範例假設資料檔案的格式與所顯示的樣本資料相同。在下列樣本中，空格充當預留位置，讓所有欄都是規格所指明的相同寬度：

```
1  Toyota Park              Bridgeview  IL0
2  Columbus Crew Stadium    Columbus    OH0
3  RFK Stadium              Washington  DC0
4  CommunityAmerica BallparkKansas City KS0
5  Gillette Stadium         Foxborough  MA68756
```

從 CSV 檔案載入 CATEGORY

假設您想要將下表所示的值載入 CATEGORY。

catid	catgroup	catname	catdesc
12	Shows	Musicals	Musical theatre
13	Shows	Plays	All "non-musical" theatre
14	Shows	Opera	All opera, light, and "rock" opera

catid	catgroup	catname	catdesc
15	Concerts	Classical	All symphony, concerto, and choir concerts

下列範例顯示文字檔案的內容，欄位值以逗號分隔。

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,All "non-musical" theatre
14,Shows,Opera,All opera, light, and "rock" opera
15,Concerts,Classical,All symphony, concerto, and choir concerts
```

如果您載入此檔案時使用 DELIMITER 參數來指定逗號分隔輸入，COPY 命令會失敗，因為有些輸入欄位包含逗號。您可以使用 CSV 參數，並以引號字元括住含有逗號的欄位，即可避免此問題。如果引號括住的字串內出現引號字元，則需要多加一個引號字元才能將其逸出。預設引號字元是雙引號，所以您需要多加一個雙引號來逸出每一個雙引號。新的輸入檔案如下所示。

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,"All ""non-musical"" theatre"
14,Shows,Opera,"All opera, light, and ""rock"" opera"
15,Concerts,Classical,"All symphony, concerto, and choir concerts"
```

假設檔案名稱為 `category_csv.txt`，您可以使用下列 COPY 命令來載入檔案：

```
copy category
from 's3://mybucket/data/category_csv.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
csv;
```

或者，若要避免需要逸出輸入中的雙引號，您可以使用 QUOTE AS 參數來指定不同的引號字元。例如，下列的 `category_csv.txt` 版本使用 '%' 做為引號字元。

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,%All "non-musical" theatre%
14,Shows,Opera,%All opera, light, and "rock" opera%
15,Concerts,Classical,%All symphony, concerto, and choir concerts%
```

下列 COPY 命令使用 QUOTE AS 來載入 `category_csv.txt`：

```
copy category
from 's3://mybucket/data/category_csv.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
csv quote as '%';
```

載入 VENUE 時將明確值提供給 IDENTITY 欄

下列範例假設建立 VENUE 資料表時，至少有一欄 (例如 venueid 欄) 指定為 IDENTITY 欄。此命令覆寫 IDENTITY 預設行為，亦即自動產生 IDENTITY 欄的值，而改以從 venue.txt 檔案載入明確值。使用 EXLICIT_IDS 選項時，Amazon Redshift 不會檢查是否將重複的 IDENTITY 值載入到資料表中。

```
copy venue
from 's3://mybucket/data/venue.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
explicit_ids;
```

從縱線分隔 GZIP 檔案載入 TIME

下列範例從縱線分隔 GZIP 檔案載入 TIME 資料表：

```
copy time
from 's3://mybucket/data/timerows.gz'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
gzip
delimiter '|';
```

載入時間戳記或日期戳記

下列範例載入含有格式化時間戳記的資料。

> **Note**
>
> HH:MI:SS 的 TIMEFORMAT 也可以支援超過 SS 的小數秒，精細程度可達到微秒。此範例中使用的檔案 time.txt 包含一列，即 2009-01-12 14:15:57.119568。

```
copy timestamp1
from 's3://mybucket/data/time.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
timeformat 'YYYY-MM-DD HH:MI:SS';
```

此複製的結果如下所示：

```
select * from timestamp1;
c1
----------------------------
2009-01-12 14:15:57.119568
(1 row)
```

從含有預設值的檔案載入資料

下列範例使用一個從 TICKIT 資料庫中的 VENUE 資料表變化而來的版本。假設 VENUE_NEW 資料表是使用下列陳述式來定義：

```
create table venue_new(
venueid smallint not null,
venuename varchar(100) not null,
venuecity varchar(30),
venuestate char(2),
venueseats integer not null default '1000');
```

假設 venue_noseats.txt 資料檔案的 VENUESEATS 欄沒有值，如下列範例所示：

```
1|Toyota Park|Bridgeview|IL|
2|Columbus Crew Stadium|Columbus|OH|
3|RFK Stadium|Washington|DC|
4|CommunityAmerica Ballpark|Kansas City|KS|
5|Gillette Stadium|Foxborough|MA|
6|New York Giants Stadium|East Rutherford|NJ|
7|BMO Field|Toronto|ON|
8|The Home Depot Center|Carson|CA|
9|Dick's Sporting Goods Park|Commerce City|CO|
10|Pizza Hut Park|Frisco|TX|
```

下列 COPY 陳述式可成功從檔案載入資料表，並將 DEFAULT 值 ('1000') 套用至省略的欄：

```
copy venue_new(venueid, venuename, venuecity, venuestate)
from 's3://mybucket/data/venue_noseats.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

現在檢視已載入的資料表：

```
select * from venue_new order by venueid;
venueid |         venuename          |    venuecity    | venuestate | venueseats
---------+----------------------------+-----------------+------------+------------
1 | Toyota Park                | Bridgeview      | IL         |       1000
2 | Columbus Crew Stadium      | Columbus        | OH         |       1000
3 | RFK Stadium                | Washington      | DC         |       1000
4 | CommunityAmerica Ballpark  | Kansas City     | KS         |       1000
5 | Gillette Stadium           | Foxborough      | MA         |       1000
6 | New York Giants Stadium    | East Rutherford | NJ         |       1000
7 | BMO Field                  | Toronto         | ON         |       1000
8 | The Home Depot Center      | Carson          | CA         |       1000
9 | Dick's Sporting Goods Park | Commerce City   | CO         |       1000
10 | Pizza Hut Park             | Frisco          | TX         |       1000
(10 rows)
```

在下列範例中，除了假設檔案不含 VENUESEATS 資料，也假設不含 VENUENAME 資料：

```
1||Bridgeview|IL|
2||Columbus|OH|
3||Washington|DC|
4||Kansas City|KS|
5||Foxborough|MA|
6||East Rutherford|NJ|
7||Toronto|ON|
8||Carson|CA|
9||Commerce City|CO|
10||Frisco|TX|
```

使用同樣的資料表定義，下列 COPY 陳述式會失敗，因為未指定 VENUENAME 的 DEFAULT 值，且 VENUENAME 是 NOT NULL 欄：

```
copy venue(venueid, venuecity, venuestate)
from 's3://mybucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

現在假設一個使用 IDENTITY 欄的變化版 VENUE 資料表：

```
create table venue_identity(
```

```
venueid int identity(1,1),
venuename varchar(100) not null,
venuecity varchar(30),
venuestate char(2),
venueseats integer not null default '1000');
```

如同上述範例，假設 VENUESEATS 欄在來源檔案中沒有對應的值。下列 COPY 陳述式可成功載入資料表，包括預先定義的 IDENTITY 資料值，而不是自動產生那些值：

```
copy venue(venueid, venuename, venuecity, venuestate)
from 's3://mybucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' explicit_ids;
```

此陳述式會失敗，因為不含 IDENTITY 欄 (欄清單中缺少 VENUEID)，卻包含 EXPLICIT_IDS 參數：

```
copy venue(venuename, venuecity, venuestate)
from 's3://mybucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' explicit_ids;
```

此陳述式會失敗，因為不含 EXPLICIT_IDS 參數：

```
copy venue(venueid, venuename, venuecity, venuestate)
from 's3://mybucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

搭配 ESCAPE 選項來 COPY 資料

下列範例示範如何載入符合分隔符號字元 (在此例子中是縱線字元) 的字元。在輸入檔案中，請確定您要載入的所有縱線字元 (|) 都以反斜線字元 (\) 逸出。載入以 ESCAPE 參數載入檔案。

```
$ more redshiftinfo.txt
1|public\|event\|dwuser
2|public\|sales\|dwuser

create table redshiftinfo(infoid int,tableinfo varchar(50));

copy redshiftinfo from 's3://mybucket/data/redshiftinfo.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
delimiter '|' escape;

select * from redshiftinfo order by 1;
infoid |       tableinfo
-------+--------------------
1      | public|event|dwuser
2      | public|sales|dwuser
(2 rows)
```

如果不指定 ESCAPE 參數，此 COPY 命令會失敗並傳回 `Extra column(s) found` 錯誤。

> **Important**
>
> 如果您使用 COPY 搭配 ESCAPE 參數來載入資料，則在 UNLOAD 命令中也必須指定 ESCAPE 參數，以產生對等的輸出檔案。同樣地，如果使用 ESCAPE 參數來 UNLOAD，則在 COPY 相同的資料時需要使用 ESCAPE。

從 JSON 複製的範例

在下列範例中，您會將下列資料載入 CATEGORY 資料表。

CATID	CATGROUP	CATNAME	CATDESC
1	運動	MLB	美國職棒大聯盟
2	運動	NHL	National Hockey League
3	運動	NFL	National Football League
4	運動	NBA	National Basketball Association
5	Concerts	Classical	All symphony, concerto, and choir concerts

主題

- 使用 'auto' 選項從 JSON 資料載入
- 使用 'auto ignorecase' 選項從 JSON 資料載入
- 使用 JSONPaths 檔案從 JSON 資料載入

- 使用 JSONPaths 檔案從 JSON 陣列載入

使用 'auto' 選項從 JSON 資料載入

若要使用 'auto' 選項從 JSON 資料載入，JSON 資料必須包含一組物件。金鑰名稱必須符合欄名稱，但順序並不重要。以下顯示一個名為 category_object_auto.json 之檔案的內容。

```
{
    "catdesc": "Major League Baseball",
    "catid": 1,
    "catgroup": "Sports",
    "catname": "MLB"
}
{
    "catgroup": "Sports",
    "catid": 2,
    "catname": "NHL",
    "catdesc": "National Hockey League"
}{
    "catid": 3,
    "catname": "NFL",
    "catgroup": "Sports",
    "catdesc": "National Football League"
}
{
    "bogus": "Bogus Sports LLC",
    "catid": 4,
    "catgroup": "Sports",
    "catname": "NBA",
    "catdesc": "National Basketball Association"
}
{
    "catid": 5,
    "catgroup": "Shows",
    "catname": "Musicals",
    "catdesc": "All symphony, concerto, and choir concerts"
}
```

若要從上述範例的 JSON 資料檔案載入，請執行下列 COPY 命令。

```
copy category
from 's3://mybucket/category_object_auto.json'
```

```
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 'auto';
```

使用 'auto ignorecase' 選項從 JSON 資料載入

若要使用 'auto ignorecase' 選項從 JSON 資料載入，JSON 資料必須包含一組物件。索引鍵名稱的大小寫不必符合欄名稱，順序也不重要。以下顯示一個名為 category_object_auto-ignorecase.json 之檔案的內容。

```
{
    "CatDesc": "Major League Baseball",
    "CatID": 1,
    "CatGroup": "Sports",
    "CatName": "MLB"
}
{
    "CatGroup": "Sports",
    "CatID": 2,
    "CatName": "NHL",
    "CatDesc": "National Hockey League"
}{
    "CatID": 3,
    "CatName": "NFL",
    "CatGroup": "Sports",
    "CatDesc": "National Football League"
}
{
    "bogus": "Bogus Sports LLC",
    "CatID": 4,
    "CatGroup": "Sports",
    "CatName": "NBA",
    "CatDesc": "National Basketball Association"
}
{
    "CatID": 5,
    "CatGroup": "Shows",
    "CatName": "Musicals",
    "CatDesc": "All symphony, concerto, and choir concerts"
}
```

若要從上述範例的 JSON 資料檔案載入，請執行下列 COPY 命令。

```
copy category
```

```
from 's3://mybucket/category_object_auto ignorecase.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 'auto ignorecase';
```

使用 JSONPaths 檔案從 JSON 資料載入

如果 JSON 資料物件沒有直接對應至欄名稱，您可以使用 JSONPaths 檔案將 JSON 元素映射至欄。JSON 來源資料中的順序並不重要，但 JSONPaths 檔案運算式的順序必須符合欄順序。假設您有下列資料檔案，名稱為 category_object_paths.json。

```
{
    "one": 1,
    "two": "Sports",
    "three": "MLB",
    "four": "Major League Baseball"
}
{
    "three": "NHL",
    "four": "National Hockey League",
    "one": 2,
    "two": "Sports"
}
{
    "two": "Sports",
    "three": "NFL",
    "one": 3,
    "four": "National Football League"
}
{
    "one": 4,
    "two": "Sports",
    "three": "NBA",
    "four": "National Basketball Association"
}
{
    "one": 6,
    "two": "Shows",
    "three": "Musicals",
    "four": "All symphony, concerto, and choir concerts"
}
```

下列 JSONPaths 檔案 (名為 category_jsonpath.json) 將來源資料映射至資料表欄。

```
{
    "jsonpaths": [
        "$['one']",
        "$['two']",
        "$['three']",
        "$['four']"
    ]
}
```

若要從上述範例的 JSON 資料檔案載入，請執行下列 COPY 命令。

```
copy category
from 's3://mybucket/category_object_paths.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 's3://mybucket/category_jsonpath.json';
```

使用 JSONPaths 檔案從 JSON 陣列載入

若要從包含一組陣列的 JSON 資料載入，您必須使用 JSONPaths 檔案將陣列元素映射至欄。假設您有下列資料檔案，名稱為 `category_array_data.json`。

```
[1,"Sports","MLB","Major League Baseball"]
[2,"Sports","NHL","National Hockey League"]
[3,"Sports","NFL","National Football League"]
[4,"Sports","NBA","National Basketball Association"]
[5,"Concerts","Classical","All symphony, concerto, and choir concerts"]
```

下列 JSONPaths 檔案 (名為 `category_array_jsonpath.json`) 將來源資料映射至資料表欄。

```
{
    "jsonpaths": [
        "$[0]",
        "$[1]",
        "$[2]",
        "$[3]"
    ]
}
```

若要從上述範例的 JSON 資料檔案載入，請執行下列 COPY 命令。

```
copy category
```

```
from 's3://mybucket/category_array_data.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 's3://mybucket/category_array_jsonpath.json';
```

從 Avro 複製的範例

在下列範例中，您會將下列資料載入 CATEGORY 資料表。

CATID	CATGROUP	CATNAME	CATDESC
1	運動	MLB	美國職棒大聯盟
2	運動	NHL	National Hockey League
3	運動	NFL	National Football League
4	運動	NBA	National Basketball Association
5	Concerts	Classical	All symphony, concerto, and choir concerts

主題

- 使用 'auto' 選項從 Avro 資料載入
- 使用 'auto ignorecase' 選項從 Avro 資料載入
- 使用 JSONPaths 檔案從 Avro 資料載入

使用 'auto' 選項從 Avro 資料載入

若要使用 `'auto'` 引數從 Avro 資料載入，Avro 結構描述中的欄位名稱必須符合欄名稱。使用 `'auto'` 引數時，順序並不重要。以下顯示一個名為 `category_auto.avro` 之檔案的結構描述。

```
{
    "name": "category",
    "type": "record",
    "fields": [
        {"name": "catid", "type": "int"},
        {"name": "catdesc", "type": "string"},
        {"name": "catname", "type": "string"},
        {"name": "catgroup", "type": "string"},
```

```
}
```

Avro 檔案中的資料是二進位格式，無法直接閱讀。以下顯示 category_auto.avro 檔案中之資料的 JSON 表示法。

```
{
   "catid": 1,
   "catdesc": "Major League Baseball",
   "catname": "MLB",
   "catgroup": "Sports"
}
{
   "catid": 2,
   "catdesc": "National Hockey League",
   "catname": "NHL",
   "catgroup": "Sports"
}
{
   "catid": 3,
   "catdesc": "National Basketball Association",
   "catname": "NBA",
   "catgroup": "Sports"
}
{
   "catid": 4,
   "catdesc": "All symphony, concerto, and choir concerts",
   "catname": "Classical",
   "catgroup": "Concerts"
}
```

若要從上述範例的 Avro 資料檔案載入，請執行下列 COPY 命令。

```
copy category
from 's3://mybucket/category_auto.avro'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as avro 'auto';
```

使用 'auto ignorecase' 選項從 Avro 資料載入

若要使用 'auto ignorecase' 引數從 Avro 資料載入，Avro 結構描述中欄位名稱的大小寫不必符合欄名稱的大小寫。使用 'auto ignorecase' 引數時，順序並不重要。以下顯示一個名為 category_auto-ignorecase.avro 之檔案的結構描述。

```
{
    "name": "category",
    "type": "record",
    "fields": [
        {"name": "CatID", "type": "int"},
        {"name": "CatDesc", "type": "string"},
        {"name": "CatName", "type": "string"},
        {"name": "CatGroup", "type": "string"},
}
```

Avro 檔案中的資料是二進位格式，無法直接閱讀。以下顯示 category_auto-ignorecase.avro 檔案中之資料的 JSON 表示法。

```
{
   "CatID": 1,
   "CatDesc": "Major League Baseball",
   "CatName": "MLB",
   "CatGroup": "Sports"
}
{
   "CatID": 2,
   "CatDesc": "National Hockey League",
   "CatName": "NHL",
   "CatGroup": "Sports"
}
{
   "CatID": 3,
   "CatDesc": "National Basketball Association",
   "CatName": "NBA",
   "CatGroup": "Sports"
}
{
   "CatID": 4,
   "CatDesc": "All symphony, concerto, and choir concerts",
   "CatName": "Classical",
   "CatGroup": "Concerts"
}
```

若要從上述範例的 Avro 資料檔案載入，請執行下列 COPY 命令。

```
copy category
from 's3://mybucket/category_auto-ignorecase.avro'
```

```
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as avro 'auto ignorecase';
```

使用 JSONPaths 檔案從 Avro 資料載入

如果 Avro 結構描述中的欄位名稱沒有直接對應至欄名稱，您可以使用 JSONPaths 檔案將結構描述元素映射至欄。JSONPaths 檔案表達式的順序必須符合欄順序。

假設您有一個名為 `category_paths.avro` 的資料檔案，其中包含的資料與上述範例相同，但使用下列結構描述。

```
{
    "name": "category",
    "type": "record",
    "fields": [
        {"name": "id", "type": "int"},
        {"name": "desc", "type": "string"},
        {"name": "name", "type": "string"},
        {"name": "group", "type": "string"},
        {"name": "region", "type": "string"}
     ]
}
```

下列 JSONPaths 檔案 (名為 `category_path.avropath`) 將來源資料映射至資料表欄。

```
{
    "jsonpaths": [
        "$['id']",
        "$['group']",
        "$['name']",
        "$['desc']"
    ]
}
```

若要從上述範例的 Avro 資料檔案載入，請執行下列 COPY 命令。

```
copy category
from 's3://mybucket/category_object_paths.avro'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format avro 's3://mybucket/category_path.avropath ';
```

準備要搭配 ESCAPE 選項來 COPY 的檔案

下列範例描述在使用 COPY 命令搭配 ESCAPE 參數，將資料匯入 Amazon Redshift 資料表之前，如何準備資料來「逸出」換行字元。如果不準備資料來分隔換行字元，當您執行 COPY 命令時，Amazon Redshift 會傳回載入錯誤，因為換行字元通常做為記錄分隔符號。

例如，假設您想要將一個檔案或外部資料表中的一欄複製到 Amazon Redshift 資料表。如果此檔案或欄包含 XML 格式的內容或類似資料，您必須確定內容中的所有換行字元 (\n) 都以反斜線字元 (\) 逸出。

包含內嵌換行字元的檔案或資料表可提供相當簡單的比對模式。每一個內嵌的換行字元很可能都接在 > 字元後面，且之間可能有幾個空格字元 (' ' 或 Tab 字元)，如下列範例所示 (在名為 nlTest1.txt 的文字檔案中)。

```
$ cat nlTest1.txt
<xml start>
<newline characters provide>
<line breaks at the end of each>
<line in content>
</xml>|1000
<xml>
</xml>|2000
```

在下列範例中，您可以執行文字處理公用程式來預先處理來源檔案，並在需要的地方插入逸出字元。(當欄資料複製到 Amazon Redshift 資料表時，會將 | 字元主要做為分隔符號來分隔欄資料。)

```
$ sed -e ':a;N;$!ba;s/>[[:space:]]*\n/>\\\n/g' nlTest1.txt > nlTest2.txt
```

同樣地，您可以使用 Perl 來執行類似的操作：

```
cat nlTest1.txt | perl -p -e 's/>\s*\n/>\\\n/g' > nlTest2.txt
```

為了方便將 nlTest2.txt 檔案中的資料載入 Amazon Redshift，我們在 Amazon Redshift 中建立一個兩欄資料表。第一欄 c1 是字元欄，將存放來自 nlTest2.txt 檔案的 XML 格式內容。第二欄 c2 存放從同一個檔案載入的整數值。

執行 sed 命令之後，您就可以使用 ESCAPE 參數，將 nlTest2.txt 檔案中的資料正確載入 Amazon Redshift 資料表。

> **Note**
>
> 在 COPY 命令中包含 ESCAPE 參數時，可逸出一些含有反斜線字元的特殊字元 (包括換行字元)。

```
copy t2 from 's3://mybucket/data/nlTest2.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
escape
delimiter as '|';

select * from t2 order by 2;

c1           |  c2
-------------+------
<xml start>
<newline characters provide>
<line breaks at the end of each>
<line in content>
</xml>
| 1000
<xml>
</xml>       | 2000
(2 rows)
```

您可以用類似的方法來準備從外部資料庫匯出的資料檔案。例如，假設是 Oracle 資料庫，在您要複製到 Amazon Redshift 的資料表中，您可以對每個受影響的欄使用 REPLACE 函數。

```
SELECT c1, REPLACE(c2, \n',\\n' ) as c2 from my_table_with_xml
```

此外，許多資料庫匯出及擷取、轉換、載入 (ETL) 的工具 (經常處理大量資料)，都提供選項來指定逸出和分隔符號字元。

將 Shapefile 載入 Amazon Redshift

下列範例示範如何使用 COPY 來載入 Esri Shapefile。如需載入 Shapefile 的相關資訊，請參閱將 Shapefile 載入 Amazon Redshift。

載入 Shapefile

下列步驟說明如何使用 COPY 命令從 Amazon S3 擷取 OpenStreetMap 資料。此範例假設 Geofabrik 下載網站中的挪威 shapefile 存檔已上傳到您區域中的私有 Amazon S3 儲存貯體。 AWS `.shp`、`.shx` 和 `.dbf` 檔案必須共用相同的 Amazon S3 字首和檔案名稱。

不使用簡化擷取資料

以下命令會建立資料表並擷取資料，這些資料無需任何簡化即可符合最大幾何大小。在您偏好的 GIS 軟體中開啟 `gis_osm_natural_free_1.shp`，並檢查此圖層中的欄。依預設，IDENTITY 或 GEOMETRY 欄都是第一個。當 GEOMETRY 欄是第一個，你可以建立資料表，如下所示。

```
CREATE TABLE norway_natural (
   wkb_geometry GEOMETRY,
   osm_id BIGINT,
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

或者，當 IDENTITY 欄是第一個，你可以建立資料表，如下所示。

```
CREATE TABLE norway_natural_with_id (
   fid INT IDENTITY(1,1),
   wkb_geometry GEOMETRY,
   osm_id BIGINT,
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

現在，您可以使用 COPY 擷取資料。

```
COPY norway_natural FROM 's3://bucket_name/shapefiles/norway/
gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural' completed, 83891 record(s) loaded successfully
```

或者，您可以如下所示擷取資料。

```
COPY norway_natural_with_id FROM 's3://bucket_name/shapefiles/norway/
gis_osm_natural_free_1.shp'
```

```
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural_with_id' completed, 83891 record(s) loaded
 successfully.
```

使用簡化擷取資料

下列命令會建立資料表並嘗試擷取資料，這些資料若不經簡化就無法符合最大幾何大小。檢查 gis_osm_water_a_free_1.shp Shapefile 並建立適當資料表，如下所示。

```
CREATE TABLE norway_water (
   wkb_geometry GEOMETRY,
   osm_id BIGINT,
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

COPY 命令執行時，會導致錯誤。

```
COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
ERROR:  Load into table 'norway_water' failed.  Check 'stl_load_errors' system table
 for details.
```

查詢 STL_LOAD_ERRORS 顯示幾何太大。

```
SELECT line_number, btrim(colname), btrim(err_reason) FROM stl_load_errors WHERE query
 = pg_last_copy_id();
 line_number |    btrim     |                                  btrim
-------------+--------------
+-------------------------------------------------------------------------
     1184705 | wkb_geometry | Geometry size: 1513736 is larger than maximum supported
 size: 1048447
```

為了解決這個問題，請將 SIMPLIFY AUTO 參數加入 COPY 命令以簡化幾何。

```
COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
SIMPLIFY AUTO
```

```
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';

INFO:  Load into table 'norway_water' completed, 1989196 record(s) loaded successfully.
```

若要檢視已簡化的列和幾何，請查詢 SVL_SPATIAL_SIMPLIFY。

```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
 query | line_number | maximum_tolerance | initial_size | simplified | final_size |
 final_tolerance
-------+-------------+-------------------+--------------+------------+------------
+----------------------
    20 |     1184704 |                -1 |      1513736 | t          |    1008808 |
 1.276386653895e-05
    20 |     1664115 |                -1 |      1233456 | t          |    1023584 |
 6.11707814796635e-06
```

使用 SIMPLIFY AUTO max_tolerance 且公差低於自動計算的公差，可能會導致擷取錯誤。在這種情況下，請使用 MAXERROR 忽略錯誤。

```
COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
SIMPLIFY AUTO 1.1E-05
MAXERROR 2
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';

INFO:  Load into table 'norway_water' completed, 1989195 record(s) loaded successfully.
INFO:  Load into table 'norway_water' completed, 1 record(s) could not be loaded.
 Check 'stl_load_errors' system table for details.
```

再次查詢 SVL_SPATIAL_SIMPLIFY 以識別 COPY 未能載入的記錄。

```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
 query | line_number | maximum_tolerance | initial_size | simplified | final_size |
 final_tolerance
-------+-------------+-------------------+--------------+------------+------------
+-----------------
    29 |     1184704 |           1.1e-05 |      1513736 | f          |          0 |
          0
    29 |     1664115 |           1.1e-05 |      1233456 | t          |     794432 |
     1.1e-05
```

在這個範例中，第一筆記錄無法容納，因此 simplified 欄顯示為 false。第二筆記錄在給定的公差範圍內載入。但是，最終大小大於使用自動計算的公差而不指定最大公差。

從壓縮的 Shapefile 載入

Amazon Redshift COPY 支援從壓縮的 Shapefile 中擷取資料。所有 Shapefile 元件必須具有相同的 Amazon S3 字首和相同的壓縮字尾。例如，假設您想要從上述範例中載入資料。在此情況下，gis_osm_water_a_free_1.shp.gz、gis_osm_water_a_free_1.dbf.gz 和 gis_osm_water_a_free_1.shx.gz 檔案必須共用相同的 Amazon S3 目錄。COPY 命令需要 GZIP 選項，並且 FROM 子句必須指定正確的壓縮檔案，如下所示。

```
COPY norway_natural FROM 's3://bucket_name/shapefiles/norway/compressed/
gis_osm_natural_free_1.shp.gz'
FORMAT SHAPEFILE
GZIP
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO:  Load into table 'norway_natural' completed, 83891 record(s) loaded successfully.
```

以不同的欄順序將資料載入到資料表中

如果您的資料表沒有將 GEOMETRY 作為第一欄，則可以使用欄映射將欄映射到目標資料表。例如，建立一個將 osm_id 指定為第一欄的資料表。

```
CREATE TABLE norway_natural_order (
   osm_id BIGINT,
   wkb_geometry GEOMETRY,
   code INT,
   fclass VARCHAR,
   name VARCHAR);
```

然後使用欄映射擷取 Shapefile。

```
COPY norway_natural_order(wkb_geometry, osm_id, code, fclass, name)
FROM 's3://bucket_name/shapefiles/norway/gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO:  Load into table 'norway_natural_order' completed, 83891 record(s) loaded
 successfully.
```

將資料載入至具有地理欄的資料表

如果您有一個包含 GEOGRAPHY 欄的資料表，則首先擷取至 GEOMETRY 欄中，然後將物件轉換為 GEOGRAPHY 物件。例如，將 Shapefile 複製到 GEOMETRY 欄之後，請更改資料表以新增 GEOGRAPHY 資料類型的欄。

```
ALTER TABLE norway_natural ADD COLUMN wkb_geography GEOGRAPHY;
```

然後將幾何轉換為地理。

```
UPDATE norway_natural SET wkb_geography = wkb_geometry::geography;
```

您也可以選擇捨棄 GEOMETRY 欄。

```
ALTER TABLE norway_natural DROP COLUMN wkb_geometry;
```

具有 NOLOAD 選項的 COPY 命令

若要在實際載入資料之前先驗證資料檔案，請使用 NOLOAD 選項搭配 COPY 命令。Amazon Redshift 會剖析輸入檔案，並顯示發生的任何錯誤。下列範例使用 NOLOAD 選項，而且沒有列實際載入資料表中。

```
COPY public.zipcode1
FROM 's3://mybucket/mydata/zipcode.csv'
DELIMITER ';'
IGNOREHEADER 1 REGION 'us-east-1'
NOLOAD
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/myRedshiftRole';

Warnings:
Load into table 'zipcode1' completed, 0 record(s) loaded successfully.
```

CREATE DATABASE

建立新的資料庫。

若要建立資料庫，您必須是超級使用者或具有 CREATEDB 權限。若要建立與零 ETL 整合相關聯的資料庫，您必須是超級使用者，或同時具有 CREATEDB 和 CREATEUSER 權限。

您無法在交易區塊 (BEGIN ... END) 中執行 CREATE DATABASE。如需交易的相關資訊，請參閱 可序列化隔離。

語法

```
CREATE DATABASE database_name
[ { [ WITH ]
    [ OWNER [=] db_owner ]
    [ CONNECTION LIMIT { limit | UNLIMITED } ]
    [ COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE } ]
    [ ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT } ]
  }
  | { [ WITH PERMISSIONS ] FROM DATASHARE datashare_name ] OF [ ACCOUNT account_id ]
 NAMESPACE namespace_guid }
  | { FROM { { ARN '<arn>' } { WITH DATA CATALOG SCHEMA '<schema>' | WITH NO DATA
 CATALOG SCHEMA } }
             | { INTEGRATION '<integration_id>'} }
  | { IAM_ROLE  {default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>' } }
```

參數

database_name

新資料庫的名稱。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

WITH

選用的關鍵字。

OWNER

指定資料庫擁有者。

=

選用字元。

db_owner

資料庫擁有者的使用者名稱。

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字可允許同時連線的最大數目。另外也可能限制每位使用者的連線數目。如需詳細資訊，請參閱

CREATE USER。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 STV_SESSIONS 系統畫面。

> **Note**
>
> 如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }

指定字串搜尋或比較是 CASE_SENSITIVE 或 CASE_INSENSITIVE 的子句。預設值為 CASE_SENSITIVE。

ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT }

指定對資料庫執行查詢時所用隔離層級的子句。

- 可序列化隔離 — 為並行交易提供完整的序列化能力。如需詳細資訊，請參閱 可序列化隔離。
- SNAPSHOT 隔離 — 提供隔離層級，防止更新和刪除衝突。這是在佈建叢集或無伺服器命名空間中建立之資料庫的預設值。

您可以檢式資料庫正在執行的並行模型，如下所示：

- 查詢 STV_DB_ISOLATION_LEVEL 目錄檢視。如需詳細資訊，請參閱 STV_DB_ISOLATION_LEVEL。

```
SELECT * FROM stv_db_isolation_level;
```

- 查詢 PG_DATABASE_INFO 檢視。

```
SELECT datname, datconfig FROM pg_database_info;
```

每個資料庫的隔離層級都會顯示在 concurrency_model 索引鍵旁邊。值為 1 表示 SNAPSHOT。值為 2 表示 SERIALIZABLE。

在 Amazon Redshift 資料庫中，SERIALIZABLE 和 SNAPSHOT 隔離都是可序列化隔離層級的類型。也就是說，根據 SQL 標準，不允許已變更讀取、不可重複讀取和幽靈讀取。這兩個隔離層級可保證在交易開始時，交易可在存在之資料的快照上操作，而且其他交易都不能變更該快照。但是，SNAPSHOT 隔離不提供完整的序列化，因為其不會防止在不同的資料表資料列上發生寫入偏差插入和更新。

下列案例說明使用 SNAPSHOT 隔離層級的寫入偏差更新。名為 Numbers 的資料表包含名為 digits 的資料欄，其中包含 0 和 1 值。每個使用者的 UPDATE 陳述式都不會與其他使用者重疊。但是，0 和 1 值會交換。他們執行的 SQL 會遵循此時間表，結果如下：

時間	使用者 1 動作	使用者 2 動作
1	BEGIN;	
2		BEGIN;

3	SELECT * FROM Numbers;
	`digits`
	`-`
	`-----`
	`0`
	`1`	

4		SELECT * FROM Numbers;
		`digits`
		`------`
		`0`
		`1`

5	UPDATE Numbers SET digits=0 WHERE digits=1;	

時間	使用者 1 動作	使用者 2 動作

6	SELECT * FROM Numbers
	`digits`
	`-`
	`-----`
	`0`
	`0`	

7	COMMIT	
8		Update Numbers SET digits=1 WHERE digits=0;

9		SELECT * FROM Numbers;
		`digits`
		`------`
		`1`
		`1`

10		COMMIT;

時間	使用者 1 動作	使用者 2 動作
11	SELECT * FROM Numbers; digits ----- 1 0	
12		SELECT * FROM Numbers; digits ------ 1 0

如果使用可序列化隔離執行相同的案例，Amazon Redshift 會因為可序列化違規而終止使用者 2，並傳回錯誤 1023。如需詳細資訊，請參閱 如何修正可序列化隔離錯誤。在這種情況下，只有使用者 1 可以成功提交。並非所有工作負載都需要要求可序列化隔離，在這種情況下，快照隔離就足以成為資料庫的目標隔離層級。

<ARN>從 ARN "

用來建立 AWS Glue 資料庫的資料庫 ARN。

{資料目錄綱要 '<schema>' | 沒有資料目錄資料架構}

> **Note**
>
> 只有當您的 CREATE DATABASE 命令也使用 FROM ARN 參數時，此參數才適用。

指定是否使用結構描述建立資料庫，以協助存取 AWS Glue Data Catalog中的物件。

從整合 '<integration_id>'

指定是否使用零 ETL 整合識別碼來建立資料庫。您可以`integration_id`從 SVV_ 集成系統視圖中檢索。如需範例，請參閱建立資料庫以接收零 ETL 整合的結果。如需使用零 ETL 整合建立資料庫的詳細資訊，請參閱 Amazon Redshift 管理指南中的在 Amazon Redshift 中建立目的地資料庫。

IAM_ROLE { default | 'SESSION' | 'arn:aws:iam::*<AWS ##-id>*:role/*<role-name>*' }

> **Note**
>
> 只有當您的 CREATE DATABASE 命令也使用 FROM ARN 參數時，此參數才適用。

如果您在執行 CREATE DATABASE 命令時指定與叢集關聯的 IAM 角色，當您在資料庫上執行查詢時，Amazon Redshift 將會使用該角色的登入資料。

指定 `default` 關鍵字表示使用設定為預設值且與叢集相關聯的 IAM 角色。

如果您使用聯合身分連線到 Amazon Redshift 叢集，並從使用此命令建立的外部結構描述存取資料表，請使用 `'SESSION'`。如需使用聯合身分的範例，請參閱使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取，其中會說明如何設定聯合身分。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。IAM 角色最少須具有在所要存取的 Amazon S3 儲存貯體上執行 LIST 操作，以及在儲存貯體包含的 Amazon S3 物件上執行 GET 操作的許可。若要深入了解如何在使用資料庫建立資料庫時使用 IAM_ROLE，請參閱以 AWS Glue Data Catalog 取用者身分使用 Lake 格式化管理的資料庫。

以下顯示單一 ARN 的 IAM_ROLE 參數字串語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

您可以鏈結角色，以便您的叢集可以擔任其他 IAM 角色 (可能屬於其他帳戶)。您最多可以鏈結 10 個角色。如需詳細資訊，請參閱 在 Amazon Redshift Spectrum 中鏈結 IAM 角色。

對於此 IAM 角色，請附加與以下內容相似的 IAM 許可政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-
rds-secret-VNenFy"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        }
    ]
}
```

如需建立 IAM 角色以搭配聯合查詢使用的步驟，請參閱建立秘密和 IAM 角色來使用聯合查詢。

> **Note**
>
> 不要在鏈結的角色清單中包含空格。

以下顯示鏈結三個角色的語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-
account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

搭配資料共用使用 CREATE DATABASE 的語法

下列語法描述用來從資料清單建立資料庫的 CREATE DATABASE 命令，以便在同 AWS 一帳戶內共用資料。

```
CREATE DATABASE database_name
```

```
[ [ WITH PERMISSIONS ] FROM DATASHARE datashare_name ] OF [ ACCOUNT account_id ]
 NAMESPACE namespace_guid
```

下列語法說明用來從資料清單建立資料庫的 CREATE DATABASE 命令，以便跨 AWS 帳戶共用資料。

```
CREATE DATABASE database_name
[ [ WITH PERMISSIONS ] FROM DATASHARE datashare_name ] OF ACCOUNT account_id
 NAMESPACE namespace_guid
```

搭配資料共用使用 CREATE DATABASE 的參數

FROM DATASHARE

指出資料共用所在位置的關鍵字。

datashare_name

建立取用者資料庫所用的資料共用名稱。

WITH PERMISSIONS

指定從資料共用建立的資料庫需要物件層級權限，才能存取個別資料庫物件。如果沒有這個子句，被授予資料庫 USAGE 權限的使用者或角色，就會自動擁有資料庫中所有資料庫物件的存取權。

NAMESPACE namespace_guid

指定資料共用所屬之生產者命名空間的值。

ACCOUNT account_id

指定資料共用所屬之生產者帳戶的值。

用於資料共用的 CREATE DATABASE 使用說明

身為資料庫超級使用者，當您使用 CREATE DATABASE 從 AWS 帳戶內的資料庫建立資料庫時，請指定「命名空間」選項。ACCOUNT 是選用選項。當您使用 CREATE DATABASE 從跨 AWS 帳戶的資料庫建立資料庫時，請指定來自生產者的帳戶和命名空間。

您只能為取用者叢集上的一個資料共用建立一個取用者資料庫。您無法建立參照相同資料共用的多個取用者資料庫。

建立資料庫 AWS Glue Data Catalog

若要使用資料庫 ARN 建立 AWS Glue 資料庫，請在「建立資料庫」指令中指定 ARN。

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA;
```

或者，您也可以為 IAM_ROLE 參數提供值。如需參數和接受值的相關資訊，請參閱參數。

以下是示範如何使用 IAM 角色從 ARN 建立資料庫的範例。

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA
 IAM_ROLE <iam-role-arn>
```

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA
 IAM_ROLE default;
```

您也可以使用 DATA CATALOG SCHEMA 建立資料庫。

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH DATA CATALOG SCHEMA
 <sample_schema> IAM_ROLE default;
```

建立資料庫以接收零 ETL 整合的結果

若要使用零 ETL 整合識別碼建立資料庫，請在建立資料庫命令`integration_id`中指定。

```
CREATE DATABASE destination_db_name FROM INTEGRATION 'integration_id';
```

例如，首先，從 SVV_ 集成中檢索集成 ID;

```
SELECT integration_id FROM SVV_INTEGRATION;
```

然後使用擷取的其中一個整合 ID 來建立接收零 ETL 整合的資料庫。

```
CREATE DATABASE sampledb FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111';
```

CREATE DATABASE 限制

Amazon Redshift 會針對資料庫強制執行以下限制：

- 每個叢集最多 60 個使用者定義的資料庫。

- 資料庫名稱最多 127 個位元組。
- 資料庫名稱不能是保留字。

資料庫定序

定序是一組規則，用於定義資料庫引擎如何比較和排序 SQL 中的字元類型資料。不區分大小寫的定序是最常用的定序。Amazon Redshift 使用不區分大小寫的定序來協助從其他資料倉儲進行移轉。透過不區分大小寫定序的原生支援，Amazon Redshift 會繼續使用重要的調整或最佳化方法，例如分佈索引鍵、排序索引鍵或範圍限制掃描。

COLLATE 子句會指定資料庫中所有 CHAR 和 VARCHAR 資料欄的預設定序。如果指定 CASE_INSENSITIVE，則所有 CHAR 或 VARCHAR 資料欄都會使用不區分大小寫的定序。如需有關定序的資訊，請參閱 定序序列。

在不區分大小寫的資料欄中插入或擷取的資料將會保留其原始大小寫。但是所有基於比較的字串操作 (包括排序和分組) 也都不區分大小寫。模式比對操作 (如 LIKE 述詞、類似和規則表達式函數) 也不區分大小寫。

下列 SQL 操作支援適用的定序語意：

- 比較運算子：=、<>、<、<=、>、>=。
- LIKE 運算子
- ORDER BY 子句
- GROUP BY 子句
- 使用字串比較的彙總函數，例如 MIN、MAX 和 LISTAGG
- 視窗函數，例如 PARTITION BY 子句與 ORDER BY 子句
- 純量函數 greatest() 和 least()、STRPOS()、REGEXP_COUNT()、REGEXP_REPLACE()、REGEXP_INSTR()、REGEXP_SUBS
- 相異子句
- UNION、INTERSECT 和 EXCEPT
- IN LIST

對於外部查詢 (包括 Amazon Redshift Spectrum 和 Aurora PostgreSQL 聯合查詢)，VARCHAR 或 CHAR 資料欄的定序會與目前的資料庫層級定序相同。

下列範例會查詢 Amazon Redshift Spectrum 資料表：

```
SELECT ci_varchar FROM spectrum.test_collation
WHERE ci_varchar = 'AMAZON';

ci_varchar
----------
amazon
Amazon
AMAZON
AmaZon
(4 rows)
```

如需如何使用資料庫定序建立資料表的資訊，請參閱 CREATE TABLE。

如需 COLLATE 函數的詳細資訊，請參閱 COLLATE 函數。

資料庫定序限制

以下是在 Amazon Redshift 中使用資料庫定序時的限制：

- 所有系統資料表或檢視 (包括 PG 目錄資料表和 Amazon Redshift 系統資料表) 都會區分大小寫。
- 當取用者資料庫和生產者資料庫具有不同的資料庫層級定序時，Amazon Redshift 不支援跨資料庫和跨叢集查詢。
- Amazon Redshift 在僅限領導節點查詢中不支援不區分大小寫的定序。

  下列範例顯示不支援的不區分大小寫查詢，以及 Amazon Redshift 傳送的錯誤：

  ```
  SELECT collate(usename, 'case_insensitive') FROM pg_user;
  ERROR:  Case insensitive collation is not supported in leader node only query.
  ```

- Amazon Redshift 不支援區分大小寫和不區分大小寫的資料欄之間進行互動，例如比較、函數、聯結或設定操作。

  下列範例顯示區分大小寫和不區分大小寫的資料行互動時的錯誤：

  ```
  CREATE TABLE test
    (ci_col varchar(10) COLLATE case_insensitive,
     cs_col varchar(10) COLLATE case_sensitive,
     cint int,
     cbigint bigint);
  ```

  ```
  SELECT ci_col = cs_col FROM test;
  ```

```
ERROR:  Query with different collations is not supported yet.
```

```
SELECT concat(ci_col, cs_col) FROM test;
ERROR:  Query with different collations is not supported yet.
```

```
SELECT ci_col FROM test UNION SELECT cs_col FROM test;
ERROR:  Query with different collations is not supported yet.
```

```
SELECT * FROM test a, test b WHERE a.ci_col = b.cs_col;
ERROR:  Query with different collations is not supported yet.
```

```
Select Coalesce(ci_col, cs_col) from test;
ERROR:  Query with different collations is not supported yet.
```

```
Select case when cint > 0 then ci_col else cs_col end from test;
ERROR:  Query with different collations is not supported yet.
```

- Amazon Redshift 不支援 SUPER 類型的定序。不支援在不區分大小寫的資料庫中建立 SUPER 資料欄，也不支援在 SUPER 和不區分大小寫的資料欄之間進行互動。

  下列範例會在不區分大小寫的資料庫中建立以 SUPER 做為資料類型的資料表：

  ```
  CREATE TABLE super_table (a super);
  ERROR: SUPER column is not supported in case insensitive database.
  ```

  下列範例會使用與 SUPER 資料進行比較的不區分大小寫字串來查詢資料：

  ```
  CREATE TABLE test_super_collation
  (s super, c varchar(10) COLLATE case_insensitive, i int);
  ```

  ```
  SELECT s = c FROM test_super_collation;
  ERROR:  Coercing from case insensitive string to SUPER is not supported.
  ```

若要讓這些查詢運作，請使用 COLLATE 函數來轉換一個資料行的定序，以比對另一個資料行。如需詳細資訊，請參閱 COLLATE 函數。

範例

建立資料庫

下列範例會建立名為 TICKIT 的資料庫，並將所有權提供給使用者 DWUSER。

```
create database tickit
with owner dwuser;
```

若要檢視有關資料庫的詳細資訊，請查詢 PG_DATABASE_INFO 目錄資料表。

```
select datname, datdba, datconnlimit
from pg_database_info
where datdba > 1;

 datname     | datdba | datconnlimit
-------------+--------+-------------
 admin       |    100 | UNLIMITED
 reports     |    100 | 100
 tickit      |    100 | 100
```

下列範例會建立名稱為 **sampledb** 且具有 SNAPSHOT 隔離層級的資料庫。

```
CREATE DATABASE sampledb ISOLATION LEVEL SNAPSHOT;
```

下列範例會從資料共用 salesshare 中建立資料庫 sales_db。

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

資料庫定序範例

建立不區分大小寫的資料庫

下列範例會建立 sampledb 資料庫、建立 T1 資料表，以及將資料插入 T1 資料表。

```
create database sampledb collate case_insensitive;
```

連線到您剛才使用 SQL 用戶端建立的新資料庫。使用 Amazon Redshift 查詢編輯器 v2 時，請在編輯器中選擇 sampledb。使用 RSQL 時，請使用如下所示的命令。

```
\connect sampledb;
```

```
CREATE TABLE T1 (
  col1 Varchar(20) distkey sortkey
);
```

```
INSERT INTO T1 VALUES ('bob'), ('john'), ('Mary'), ('JOHN'), ('Bob');
```

然後查詢會尋找包含 John 結果。

```
SELECT * FROM T1 WHERE col1 = 'John';

 col1
 ------
 john
 JOHN
(2 row)
```

以不區分大小寫的順序排序

下列範例顯示資料表 T1 的不區分大小寫排序。Bob 和 bob 或 John 和 john 的排序是不確定的，因為其在不區分大小寫的資料欄中是相等的。

```
SELECT * FROM T1 ORDER BY 1;

 col1
 ------
 bob
 Bob
 JOHN
 john
 Mary
(5 rows)
```

同樣地，下面範例顯示使用 GROUP BY 子句的不區分大小寫排序。Bob 和 bob 是同等的，屬於同一組。結果中會顯示哪一個是不確定的。

```
SELECT col1, count(*) FROM T1 GROUP BY 1;

 col1 | count
```

```
 -----+------
 Mary |  1
 bob  |  2
 JOHN |  2
(3 rows)
```

在不區分大小寫的資料欄上使用視窗函數進行查詢

下列範例會在不區分大小寫的資料欄上查詢視窗函數。

```
SELECT col1, rank() over (ORDER BY col1) FROM T1;

 col1 | rank
 -----+------
 bob  |   1
 Bob  |   1
 john |   3
 JOHN |   3
 Mary |   5
(5 rows)
```

使用 DISTINCT 關鍵字進行查詢

下列範例會使用 DISTINCT 關鍵字查詢 T1 資料表。

```
SELECT DISTINCT col1 FROM T1;

 col1
 ------
 bob
 Mary
 john
(3 rows)
```

使用 UNION 子句查詢

下列範例顯示資料表 T1 和 T2 UNION 的結果。

```
CREATE TABLE T2 AS SELECT * FROM T1;
```

```
SELECT col1 FROM T1 UNION SELECT col1 FROM T2;
```

```
 col1
 ------
 john
 bob
 Mary
(3 rows)
```

CREATE DATASHARE

在目前資料庫中建立新的資料共用。此資料共用的擁有者是 CREATE DATASHARE 命令的發行者。

Amazon Redshift 會將每個資料共用與單一 Amazon Redshift 資料庫相關聯。您只可以在資料共用中新增相關聯資料庫中的物件。您可以在同一個 Amazon Redshift 資料庫上建立多個資料共用。

如需有關資料共用的資訊，請參閱 管理資料共用工作。

若要檢視有關資料庫的資訊，請使用 SHOW DATASHARES。

所需權限

以下是 CREATE DATASHARE 所需的權限：

- 超級使用者
- 具有 CREATE DATASHARE 權限的使用者
- 資料庫擁有者

語法

```
CREATE DATASHARE datashare_name
[[SET] PUBLICACCESSIBLE [=] TRUE | FALSE ];
```

參數

datashare_name

資料共用的名稱。資料共用名稱必須是叢集命名空間中唯一的名稱。

[[SET] PUBLICACCESSIBLE]

該子句會指定資料共用是否可以共用至可公開存取的叢集。

SET PUBLICACCESSIBLE 的預設值為 FALSE。

使用須知

依預設，資料共用的擁有者只能有共用，不能有共用中的物件。

只有超級使用者和資料庫擁有者可以使用 CREATE DATASHARE，並將 ALTER 權限委派給其他使用者或群組。

範例

下列範例會建立 salesshare 資料共用。

```
CREATE DATASHARE salesshare;
```

下列範例會建立 AWS Data Exchange 管理的 demoshare 資料共用。

```
CREATE DATASHARE demoshare SET PUBLICACCESSIBLE TRUE, MANAGEDBY ADX;
```

CREATE EXTERNAL FUNCTION

根據 Amazon Redshift 建立純量使用者定義函數 (UDF)。 AWS Lambda 如需 Lambda 使用者定義函數的相關資訊，請參閱 建立純量 Lambda UDF。

所需權限

以下是 CREATE EXTERNAL FUNCTION 所需的權限：

- 超級使用者
- 具有 CREATE [OR REPLACE] EXTERNAL FUNCTION 權限的使用者

語法

```
CREATE [ OR REPLACE ] EXTERNAL FUNCTION external_fn_name ( [data_type] [, ...] )
RETURNS data_type
{ VOLATILE | STABLE }
LAMBDA 'lambda_fn_name'
IAM_ROLE { default | ‘arn:aws:iam::<AWS ##-id>:role/<role-name>’
RETRY_TIMEOUT milliseconds
MAX_BATCH_ROWS count
MAX_BATCH_SIZE size [ KB | MB ];
```

參數

OR REPLACE

該子句會指定已有相同名稱和輸入引數資料類型 (或簽章) 的函數存在時，取代現有函數。您可以將函數取代為定義一組相同資料類型的新函數。您必須是超級使用者才能取代函數。

如果您定義的函數與現有函數同名，但簽章不同，則會建立新函數。換言之，函數名稱將會過載。如需詳細資訊，請參閱 多載函數名稱。

external_fn_name

外部函數的名稱。如果您指定結構描述名稱 (例如 myschema.myfunction)，則會使用指定的結構描述建立函數。否則，函數會在目前結構描述中建立。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

我們建議您在所有 UDF 名稱前加上 f_。Amazon Redshift 會保留 UDF 名稱的 f_ 字首。透過使用 f_ 字首，您可以協助確保您的 UDF 名稱不會與目前或未來 Amazon Redshift 的任何內建 SQL 函數名稱衝突。如需詳細資訊，請參閱 命名 UDF。

data_type

輸入引數的資料類型。如需詳細資訊，請參閱 資料類型。

RETURNS data_type

函數所傳回值的資料類型。RETURNS 資料類型可以是任何標準 Amazon Redshift 資料類型。如需詳細資訊，請參閱 Python UDF 資料類型。

VOLATILE | STABLE

通知查詢最佳化工具有關函數的波動情形。

若要得到最理想的最佳化，請將函數標示為最嚴格的有效波動類別。從最低嚴格程度開始，依嚴格程度排列的波動類別如下所示：

- VOLATILE
- STABLE

VOLATILE

假設引數相同，即使是針對單一陳述式中的資料列，函數也可能在後續呼叫中傳回不同的結果。查詢最佳化工具無法假設波動函數的行為。使用波動函數的查詢必須重新評估每個輸入的函數。

STABLE

假設引數相同，函數一定會針對單一陳述式內處理的連續呼叫傳回相同結果。在不同陳述式中呼叫函數時，函數可能傳回不同結果。此類別可這麼做，所以最佳化工具可以減少在單個陳述式中調用函數的次數。

請注意，如果選擇的嚴格性對函數無效，則最佳化工具可能會有根據此嚴格性而略過某些呼叫的風險。這可能會導致不正確的結果集。

Lambda UDF 目前不支援 IMMUTABLE 子句。

LAMBDA 'lambda_fn_name'

Amazon Redshift 呼叫的函數名稱。

如需建立 AWS Lambda 函數的步驟，請參閱AWS Lambda 開發人員指南中的使用主控台建立 Lambda 函數。

如需 Lambda 函數所需權限的相關資訊，請參閱《AWS Lambda 開發人員指南》中的 AWS Lambda 權限。

IAM_ROLE { default | 'arn:aws:iam::*<AWS ##-id>*:role/*<role-name>*'

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREATE EXTERNAL FUNCTION 命令時與叢集關聯的 IAM 角色。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。CREATE EXTERNAL FUNCTION 已獲得授權，可透過此 IAM 角色調用 Lambda 函數。如果您的叢集已附加有權調用 Lambda 函數的現有 IAM 角色，則可以替換角色的 ARN。如需詳細資訊，請參閱 設定 Lambda UDF 的授權參數。

以下顯示 IAM_ROLE 參數的語法。

```
IAM_ROLE 'arn:aws:iam::aws-account-id:role/role-name'
```

RETRY_TIMEOUT milliseconds

Amazon Redshift 用於重試退避延遲的總時間 (以毫秒為單位)。

Amazon Redshift 不會針對任何失敗的查詢立即重試，而是執行退選，並在重試之間等待一段時間。然後，Amazon Redshift 會重試請求以重新執行失敗的查詢，直到所有延遲的總和等於或超過您指定的 RETRY_TIMEOUT 值為止。預設值為 20,000 毫秒。

調用 Lambda 函數時，Amazon Redshift 會針對接收到錯誤 (例如 `TooManyRequestsException`、`EC2ThrottledException` 和 `ServiceException`) 的查詢重試。

您可以將 RETRY_TIMEOUT 參數設定為 0 毫秒，以防止 Lambda UDF 進行任何重試。

MAX_BATCH_ROWS 計數

Amazon Redshift 在單一批次請求中針對單一 Lambda 調用傳送的資料列數目上限。

此參數的最小值為 1。最大值為 INT_MAX 或 2,147,483,647。

此為選用參數。預設值為 INT_MAX 或 2,147,483,647。

MAX_BATCH_SIZE size [KB | MB]

Amazon Redshift 在單一批次請求中針對單一 Lambda 調用傳送的資料承載大小上限。

此參數的最小值為 1 KB。最大值為 5 MB。

此參數的預設值為 5 MB。

您可以選擇 KB 和 MB。如果未設定測量單位，Amazon Redshift 會預設為使用 KB。

使用須知

建立 Lambda UDF 時應考慮下列事項：

- 輸入引數上的 Lambda 函數調用順序不是固定或可保證的。其可能會因執行查詢的執行個體而有所不同，具體取決於叢集組態。
- 函數不能保證一次且僅一次地套用到每個輸入引數。Amazon Redshift 之間的交互 AWS Lambda 可能會導致具有相同輸入的重複呼叫。

範例

以下是使用純量 Lambda 使用者定義函數 (UDF) 的範例。

使用 Node.js Lambda 函數的純量 Lambda UDF 範例

下列範例會建立名為 `exfunc_sum` 的外部函數，該函數會接受兩個整數做為輸入引數。該函數會傳回作為整數輸出的總和。要呼叫的 Lambda 函數名稱為 `lambda_sum`。用於此 Lambda 函數的語言是 Node.js 12.x。請務必指定 IAM 角色。此範例使用 `'arn:aws:iam::123456789012:user/johndoe'` 做為 IAM 角色。

```
CREATE EXTERNAL FUNCTION exfunc_sum(INT,INT)
RETURNS INT
VOLATILE
```

```
LAMBDA 'lambda_sum'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';
```

Lambda 函數會接受請求承載並逐一查看每一列。單列中的所有值都會加入以計算該列的總和，而這會保存在回應陣列中。結果陣列中的資料列數會與請求承載中接收的資料列數相似。

JSON 回應承載必須在「結果」欄位中包含結果資料，才能由外部函數辨識。在傳送至 Lambda 函數的請求中，引數欄位會包含資料承載。在批次處理請求的情況下，資料承載中可以有多個資料列。下列 Lambda 函數會逐一查看請求資料承載中的所有資料列。也會單獨逐一查看單一資料列中的所有值。

```
exports.handler = async (event) => {
    // The 'arguments' field in the request sent to the Lambda function contains the
 data payload.
    var t1 = event['arguments'];

    // 'len(t1)' represents the number of rows in the request payload.
    // The number of results in the response payload should be the same as the number
 of rows received.
    const resp = new Array(t1.length);

    // Iterating over all the rows in the request payload.
    for (const [i, x] of t1.entries())
    {
        var sum = 0;
        // Iterating over all the values in a single row.
        for (const y of x) {
            sum = sum + y;
        }
        resp[i] = sum;
    }
    // The 'results' field should contain the results of the lambda call.
    const response = {
        results: resp
    };
    return JSON.stringify(response);
};
```

下列範例會呼叫具有常值的外部函數。

```
select exfunc_sum(1,2);
exfunc_sum
------------
```

```
 3
(1 row)
```

下列範例會建立名為 t_sum 的資料表，其中包含整數資料類型的兩個資料欄 c1 和 c2，並插入兩列資料。然後透過傳遞此資料表的資料欄名稱來呼叫外部函數。這兩個資料表資料列會在請求承載中以批次請求的形式傳送，以做為單一 Lambda 調用。

```
CREATE TABLE t_sum(c1 int, c2 int);
INSERT INTO t_sum VALUES (4,5), (6,7);
SELECT exfunc_sum(c1,c2) FROM t_sum;
 exfunc_sum
---------------
 9
 13
(2 rows)
```

使用 RETRY_TIMEOUT 屬性的純量 Lambda UDF 範例

在下文中，您可以找到如何在 Lambda UDF 中使用 RETRY_TIMEOUT 屬性的範例。

AWS Lambda 函數具有您可以為每個函數設置的並發限制。如需並行限制的詳細資訊，請參閱AWS Lambda 開發人員指南中的管理 Lambda 函數的並行性，以及運算部落格上的管理 AWS Lambda 函數並行文章。 AWS

當 Lambda UDF 處理的要請數目超過並行限制時，新要求會收到 TooManyRequestsException 錯誤。Lambda UDF 會針對此錯誤重試，直到傳送至 Lambda 函數的請求之間的所有延遲總和等於或超過您設定的 RETRY_TIMEOUT 值為止。預設的 RETRY_TIMEOUT 值為 20,000 毫秒。

下列範例會使用名為 exfunc_sleep_3 的 Lambda 函數。該函數會接受請求承載、逐一查看每一個資料列，並將輸入轉換為大寫。然後休眠 3 秒鐘後傳回結果。用於此 Lambda 函數的語言是 Python 3.8。

結果陣列中的資料列數會與請求承載中接收的資料列數相似。JSON 回應承載必須在 results 欄位中包含結果資料，才能由外部函數辨識。在傳送至 Lambda 函數的請求中，arguments 欄位會包含資料承載。在批次處理請求的情況下，資料承載中可以有多個資料列。

此函數的並行限制會在保留並行中特別設定為 1，以示範如何使用 RETRY_TIMEOUT 屬性。當屬性設定為 1 時，Lambda 函數一次只能處理一個請求。

```
import json
import time
```

```
def lambda_handler(event, context):
    t1 = event['arguments']
    # 'len(t1)' represents the number of rows in the request payload.
    # The number of results in the response payload should be the same as the number of
 rows received.
    resp = [None]*len(t1)

    # Iterating over all rows in the request payload.
    for i, x in enumerate(t1):
        # Iterating over all the values in a single row.
        for j, y in enumerate(x):
            resp[i] = y.upper()

    time.sleep(3)
    ret = dict()
    ret['results'] = resp
    ret_json = json.dumps(ret)
    return ret_json
```

接下來，另外兩個範例會說明 RETRY_TIMEOUT 屬性。他們每個人都會調用單一 Lambda UDF。呼叫 Lambda UDF 時，每個範例都會執行相同的 SQL 查詢，以同時從兩個並行的資料庫工作階段調用 Lambda UDF。當 UDF 處理第一個調用 Lambda UDF 的查詢時，第二個查詢會收到 `TooManyRequestsException` 錯誤。之所以發生此結果，是因為您在 UDF 中特別將保留並行設定為 1。如需如何為 Lambda 函數設定保留並行的相關資訊，請參閱設定保留並行。

接下來的第一個範例會將 Lambda UDF 的 RETRY_TIMEOUT 屬性設定為 0 毫秒。如果 Lambda 請求從 Lambda 函數收到任何例外狀況，Amazon Redshift 不會進行任何重試。之所以發生這個結果，是因為 RETRY_TIMEOUT 屬性設定為 0。

```
CREATE OR REPLACE EXTERNAL FUNCTION exfunc_upper(varchar)
RETURNS varchar
VOLATILE
LAMBDA 'exfunc_sleep_3'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test'
RETRY_TIMEOUT 0;
```

將 RETRY_TIMEOUT 設定為 0 時，您可以從不同的資料庫工作階段執行下列兩個查詢，以查看不同的結果。

第一個使用 Lambda UDF 的 SQL 查詢成功執行。

```
select exfunc_upper('Varchar');
```

```
 exfunc_upper
 --------------
 VARCHAR
(1 row)
```

同時從不同資料庫工作階段執行的第二個查詢會收到 TooManyRequestsException 錯誤。

```
select exfunc_upper('Varchar');
ERROR:  Rate Exceeded.; Exception: TooManyRequestsException; ShouldRetry: 1
DETAIL:
-----------------------------------------------
error:  Rate Exceeded.; Exception: TooManyRequestsException; ShouldRetry: 1
code:      32103
context:query:     0
location:  exfunc_client.cpp:102
process:   padbmaster [pid=26384]
-----------------------------------------------
```

接下來的第二個範例會將 Lambda UDF 的 RETRY_TIMEOUT 屬性設定為 3,000 毫秒。即使第二個查詢同時執行，Lambda UDF 也會重試，直到總延遲為 3,000 毫秒為止。因此，兩個查詢都會成功執行。

```
CREATE OR REPLACE EXTERNAL FUNCTION exfunc_upper(varchar)
RETURNS varchar
VOLATILE
LAMBDA 'exfunc_sleep_3'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test'
RETRY_TIMEOUT 3000;
```

將 RETRY_TIMEOUT 設定為 3,000 毫秒時，您可以從不同的資料庫工作階段執行下列兩個查詢，以查看相同的結果。

第一個執行 Lambda UDF 的 SQL 查詢成功執行。

```
select exfunc_upper('Varchar');
 exfunc_upper
 --------------
 VARCHAR
(1 row)
```

第二個查詢會同時執行，Lambda UDF 也會重試，直到總延遲為 3,000 毫秒為止。

```
select exfunc_upper('Varchar');
 exfunc_upper
--------------
 VARCHAR
(1 row)
```

使用 Python Lambda 函數的純量 Lambda UDF 範例

下列範例會建立名為 exfunc_multiplication 且會乘以數字並傳回整數的外部函數。此範例包含 Lambda 回應中的成功和 error_msg 欄位。當乘法結果中有整數溢位，且 error_msg 訊息設定為 Integer multiplication overflow 時，成功欄位會設定為 false。exfunc_multiplication 函數採用三個整數作為輸入參數，並傳回總和作為整數輸出。

要呼叫的 Lambda 函數名稱為 lambda_multiplication。用於此 Lambda 函數的語言是 Python 3.8。請務必指定 IAM 角色。

```
CREATE EXTERNAL FUNCTION exfunc_multiplication(int, int, int)
RETURNS INT
VOLATILE
LAMBDA 'lambda_multiplication'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';
```

Lambda 函數會接受請求承載並逐一查看每一列。單列中的所有值都會相乘以計算該列的結果，而這會保存在回應清單中。此範例使用預設為 true 的布林成功值。如果資料列的乘法結果有整數溢位，則成功值會設定為 false。然後迭代循環會中斷。

建立回應承載時，如果成功值為 false，則下列 Lambda 函數會在承載中新增 error_msg 欄位。也會將錯誤訊息設定為 Integer multiplication overflow。如果成功值為 true，則結果資料會新增到結果欄位中。結果陣列中的資料列數 (如果有的話) 會與請求承載中接收的資料列數相似。

在傳送至 Lambda 函數的請求中，引數欄位會包含資料承載。在批次處理請求的情況下，資料承載中可以有多個資料列。下列 Lambda 函數會逐一查看請求資料承載中的所有資料列，並個別逐一查看單一資料列中的所有值。

```
import json
def lambda_handler(event, context):
    t1 = event['arguments']
    # 'len(t1)' represents the number of rows in the request payload.
    # The number of results in the response payload should be the same as the number of
 rows received.
    resp = [None]*len(t1)
```

```
    # By default success is set to 'True'.
    success = True
    # Iterating over all rows in the request payload.
    for i, x in enumerate(t1):
        mul = 1
        # Iterating over all the values in a single row.
        for j, y in enumerate(x):
            mul = mul*y

        # Check integer overflow.
        if (mul >= 9223372036854775807 or mul <= -9223372036854775808):
            success = False
            break
        else:
            resp[i] = mul
    ret = dict()
    ret['success'] = success
    if not success:
        ret['error_msg'] = "Integer multiplication overflow"
    else:
        ret['results'] = resp
    ret_json = json.dumps(ret)

    return ret_json
```

下列範例會呼叫具有常值的外部函數。

```
SELECT exfunc_multiplication(8, 9, 2);
  exfunc_multiplication
--------------------------
          144
(1 row)
```

下列範例會建立名為 t_multi 的資料表，其中包含整數資料類型的三個資料欄 c1、c2 和 c3。外部函數會透過傳遞此資料表的資料欄名稱來呼叫。資料會以這種方式插入，進而導致整數溢位，以顯示錯誤的傳播方式。

```
CREATE TABLE t_multi (c1 int, c2 int, c3 int);
INSERT INTO t_multi VALUES (2147483647, 2147483647, 4);
SELECT exfunc_multiplication(c1, c2, c3) FROM t_multi;
DETAIL:
```

```
  -----------------------------------------------
  error:  Integer multiplication overflow
  code:      32004context:
  context:
  query:     38
  location:  exfunc_data.cpp:276
  process:   query2_16_38 [pid=30494]
  -----------------------------------------------
```

CREATE EXTERNAL SCHEMA

在目前資料庫中建立新的外部結構描述。您可以使用此外部結構描述連線到 Amazon RDS for PostgreSQL 或 Amazon Aurora PostgreSQL 相容版本資料庫。您也可以建立外部結構描述來參考外部資料目錄 (例如 AWS Glue Athena) 中的資料庫，或是 Apache Hive 中繼存放區 (例如 Amazon EMR) 中的資料庫。

此結構描述的擁有者是 CREATE EXTERNAL SCHEMA 命令的發行者。若要轉移外部結構描述的所有權，請使用 ALTER SCHEMA 來變更擁有者。若要將結構描述的存取權授予其他使用者或使用者群組，請使用 GRANT 命令。

您無法使用 GRANT 或 REVOKE 命令處理外部資料表的許可。這時請改為在外部結構描述授予和撤銷許可。

> **Note**
>
> 如果您目前在 Amazon Athena 資料目錄中有 Redshift Spectrum 外部資料表，則可以將 Athena 資料目錄遷移到 AWS Glue Data Catalog。若要將資 AWS Glue 料目錄與 Redshift 頻譜搭配使用，您可能需要變更 AWS Identity and Access Management (IAM) 政策。如需詳細資訊，請參閱 Athena 使用指南中的升級至資 AWS Glue 料目錄。

若要檢視外部結構描述的詳細資訊，請查詢 SVV_EXTERNAL_SCHEMAS 系統畫面。

語法

以下語法描述用來使用外部資料目錄以參考資料的 CREATE EXTERNAL SCHEMA 命令。如需詳細資訊，請參閱 使用 Amazon Redshift Spectrum 查詢外部資料。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM { [ DATA CATALOG ] | HIVE METASTORE | POSTGRES | MYSQL | KINESIS | MSK |
 REDSHIFT }
```

```
[ DATABASE 'database_name' ]
[ SCHEMA 'schema_name' ]
[ REGION 'aws-region' ]
[ URI 'hive_metastore_uri' [ PORT port_number ] ]
IAM_ROLE { default | 'SESSION' | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
[ SECRET_ARN 'ssm-secret-arn' ]
[ AUTHENTICATION { none | iam } ]
[ CLUSTER_ARN 'arn:aws:kafka:<region>:<AWS ##-id>:cluster/msk/<cluster uuid>' ]
[ CATALOG_ROLE { 'SESSION' | 'catalog-role-arn-string' } ]
[ CREATE EXTERNAL DATABASE IF NOT EXISTS ]
[ CATALOG_ID 'Amazon Web Services account ID containing Glue or Lake Formation
 database' ]
```

以下語法描述用來對 RDS POSTGRES 或 Aurora PostgreSQL 使用聯合查詢以參考資料的 CREATE EXTERNAL SCHEMA 命令。您也可以建立參照串流來源的外部結構描述，例如 Kinesis Data Streams。如需詳細資訊，請參閱 使用 Amazon Redshift 中的聯合查詢來查詢資料。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM POSTGRES
DATABASE 'federated_database_name' [SCHEMA 'schema_name']
URI 'hostname' [ PORT port_number ]
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
SECRET_ARN 'ssm-secret-arn'
```

以下語法描述用來對 RDS MySQL 或 Aurora MySQL 使用聯合查詢以參考資料的 CREATE EXTERNAL SCHEMA 命令。如需詳細資訊，請參閱 使用 Amazon Redshift 中的聯合查詢來查詢資料。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM MYSQL
DATABASE 'federated_database_name'
URI 'hostname' [ PORT port_number ]
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
SECRET_ARN 'ssm-secret-arn'
```

以下語法描述用來在 Kinesis 串流中參考資料的 CREATE EXTERNAL SCHEMA 命令。如需詳細資訊，請參閱 串流擷取。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] schema_name
FROM KINESIS
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
```

下列語法描述用於參考 Amazon Managed Streaming for Apache Kafka 叢集及其主題以便從中擷取的 CREATE EXTERNAL SCHEMA 命令。CLUSTER_ARN 會指定您要從中讀取資料的 Amazon MSK 叢集。如需詳細資訊，請參閱 串流擷取。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] schema_name
FROM MSK
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
AUTHENTICATION { none | iam }
CLUSTER_ARN 'msk-cluster-arn';
```

以下語法描述用來使用跨資料庫查詢以參考資料的 CREATE EXTERNAL SCHEMA 命令。

```
CREATE EXTERNAL SCHEMA local_schema_name
FROM  REDSHIFT
DATABASE 'redshift_database_name' SCHEMA 'redshift_schema_name'
```

參數

IF NOT EXISTS

此子句會指出，若指定的結構描述已存在，則命令不應進行任何變更，且應傳回結構描述存在的訊息，而不是在發生錯誤的情況下終止。此子句在編寫指令碼時很實用，如此指令碼就不會因為 CREATE EXTERNAL SCHEMA 嘗試建立已存在的結構描述而失敗。

local_schema_name

新外部結構描述的名稱。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

FROM [DATA CATALOG] | HIVE METASTORE | POSTGRES | MYSQL | KINESIS | MSK | REDSHIFT

指出外部資料庫所在位置的關鍵字。

DATA CATALOG 會指出，外部資料庫是定義在 Athena 資料目錄中或 AWS Glue Data Catalog。

如果外部資料庫是在不同 AWS 區域的外部資料目錄中定義，則需要 REGION 參數。DATA CATALOG 是預設值。

HIVE METASTORE 指出，外部資料庫是在 Apache Hive 中繼存放區中定義。若指定了 HIVE METASTORE，則需要 URI。

POSTGRES 表示外部資料庫是在 RDS PostgreSQL 或 Aurora PostgreSQL 中定義的。

MYSQL 表示外部資料庫是在 RDS MySQL 或 Aurora MySQL 中定義的。

KINESIS 表示資料來源來自 Kinesis Data Streams。

MSK 表示資料來源是來自 Amazon MSK 的主題。

FROM REDSHIFT

指出資料庫位於 Amazon Redshift 中的關鍵字。

DATABASE 'redshift_database_name' SCHEMA 'redshift_schema_name'

Amazon Redshift 資料庫的名稱。

redshift_schema_name 表示 Amazon Redshift 中的結構描述。預設的 redshift_schema_name 為 `public`。

DATABASE 'federated_database_name'

關鍵字，指出所支援 PostgreSQL 或 MySQL 資料庫引擎中的外部資料庫名稱。

[SCHEMA 'schema_name']

schema_name 表示支援的 PostgreSQL 資料庫引擎中的結構描述。預設 schema_name 是 `public`。

當您對支援的 MySQL 資料庫引擎設定聯合查詢時，您無法指定 SCHEMA。

REGION 'aws-region'

如果外部資料庫是在 Athena 資料目錄或資料庫所在的 AWS 區域中定義的。 AWS Glue Data Catalog如果資料庫是在外部資料目錄中定義，則需要此參數。

URI 'hive_metastore_uri' [PORT port_number]

所支援 PostgreSQL 或 MySQL 資料庫引擎的主機名稱 URI 和 port_number。hostname 是複本集的前端節點。端點必須可從 Amazon Redshift 叢集連接 (可路由)。預設的 PostgreSQL port_number 為 5432。預設的 MySQL port_number 為 3306。

如果資料庫位於 Hive 中繼存放區中，請指定中繼存放區的 URI 並選擇性地指定連接埠號碼。預設連接埠號碼為 9083。

URI 不包含通訊協定規格 ("http://")。有效 URI 的範例：`uri '172.10.10.10'`。

> **Note**
>
> 支援的 PostgreSQL 或 MySQL 資料庫引擎必須位於與 Amazon Redshift 叢集相同的 VPC 中。建立連結 Amazon Redshift 和 RDS PostgreSQL 或 Aurora PostgreSQL 的安全群組。

IAM_ROLE { default | 'SESSION' | 'arn:aws:iam::*<AWS ##-id>*:role/*<role-name>*' }

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREATE EXTERNAL SCHEMA 命令時與叢集關聯的 IAM 角色。

如果您使用聯合身分連線到 Amazon Redshift 叢集，並從使用此命令建立的外部結構描述存取資料表，請使用 'SESSION'。如需詳細資訊，請參閱使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取，其中會說明如何設定聯合身分。請注意，只有在使用 DATA CATALOG 建立結構描述時，才能使用此組態 (使用 'SESSION' 取代 ARN)。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。IAM 角色最少須具有在所要存取的 Amazon S3 儲存貯體上執行 LIST 操作，以及在儲存貯體包含的 Amazon S3 物件上執行 GET 操作的許可。

以下顯示單一 ARN 的 IAM_ROLE 參數字串語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

您可以鏈結角色，以便您的叢集可以擔任其他 IAM 角色 (可能屬於其他帳戶)。您最多可以鏈結 10 個角色。如需鏈結角色的範例，請參閱 在 Amazon Redshift Spectrum 中鏈結 IAM 角色。

對於此 IAM 角色，請附加與以下內容相似的 IAM 許可政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-
rds-secret-VNenFy"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
```

```
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        }
    ]
}
```

如需建立 IAM 角色以搭配聯合查詢使用的步驟，請參閱建立秘密和 IAM 角色來使用聯合查詢。

> **Note**
>
> 不要在鏈結的角色清單中包含空格。

以下顯示鏈結三個角色的語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

SECRET_ARN 'ssm-secret-arn'

使用建立的支援 PostgreSQL 或 MySQL 資料庫引擎密碼的 Amazon 資源名稱 (ARN) AWS Secrets Manager有關如何為秘密建立和擷取 ARN 的詳細資訊，請參閱《AWS Secrets Manager 使用指南》中的建立基本秘密和擷取秘密值。

CATALOG_ROLE { 'SESSION' | catalog-role-arn-string}

使用聯合身分透過 'SESSION' 連接到 Amazon Redshift 叢集，以便對資料目錄進行身份驗證和授權。如需完成聯合身分步驟的相關資訊，請參閱使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取。請注意，只有在 DATA CATALOG 中建立結構描述時，才能使用 'SESSION' 角色。

使用叢集進行資料目錄的身分驗證和授權時所使用 IAM 角色的 Amazon Resource Name (ARN)。

如未指定 CATALOG_ROLE，則 Amazon Redshift 會使用指定的 IAM_ROLE。目錄角色必須具有存取 AWS Glue 或 Athena 中資料目錄的權限。如需詳細資訊，請參閱 Amazon Redshift Spectrum 的 IAM 政策。

以下顯示單一 ARN 的 CATALOG_ROLE 參數字串語法。

```
CATALOG_ROLE 'arn:aws:iam::<aws-account-id>:role/<catalog-role>'
```

您可以鏈結角色，以便您的叢集可以擔任其他 IAM 角色 (可能屬於其他帳戶)。您最多可以鏈結 10 個角色。如需詳細資訊，請參閱 在 Amazon Redshift Spectrum 中鏈結 IAM 角色。

> **Note**
>
> 鏈結的角色清單不得包含空格。

以下顯示鏈結三個角色的語法。

```
CATALOG_ROLE 'arn:aws:iam::<aws-account-id>:role/<catalog-role-1-name>,arn:aws:iam::<aws-account-id>:role/<catalog-role-2-name>,arn:aws:iam::<aws-account-id>:role/<catalog-role-3-name>'
```

CREATE EXTERNAL DATABASE IF NOT EXISTS

如果指定的外部資料庫不存在，此子句會使用 DATABASE 引數所指定的名稱建立外部資料庫。如果指定的外部資料庫存在，則此命令不會進行任何變更。在這種情況下，此命令會傳回外部資料庫存在的訊息，並不會因錯誤而終止。

> **Note**
>
> 您不可搭配使用 CREATE EXTERNAL DATABASE IF NOT EXISTS 與 HIVE METASTORE。
> 若要搭配為 AWS Lake Formation啟用的 Data Catalog 使用 CREATE EXTERNAL DATABASE IF NOT EXISTS，則需要 Data Catalog 的 CREATE_DATABASE 許可。

CATALOG_ID '包含 Glue 或 Lake Formation 資料庫的 Amazon Web Services 帳戶 ID帳戶 ID'

儲存資料目錄資料庫的帳戶 ID。

只有在您打算連接到 Amazon Redshift 叢集或 Amazon Redshift Serverless，並透過設定下列任一項來使用聯合身分進行資料目錄的身份驗證和授權時，才能指定 CATALOG_ID：

- CATALOG_ROLE 設定為 'SESSION'
- IAM_ROLE 設定為 'SESSION'，'CATALOG_ROLE' 設定為其預設值

如需完成聯合身分步驟的相關資訊，請參閱使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取。

AUTHENTICATION

為串流擷取定義的驗證類型。具有驗證類型的串流擷取會與 Amazon Managed Streaming for Apache Kafka 搭配運作。AUTHENTICATION 類型如下：

- none — 指定沒有驗證步驟。
- iam — 指定 IAM 身份驗證。選擇此選項時，請確保 IAM 角色具有 IAM 身份驗證的許可。如需定義外部結構描述的相關資訊，請參閱 開始使用 Amazon Managed Streaming for Apache Kafka 中的串流擷取。

CLUSTER_ARN

若是串流擷取，這是您要從中進行串流處理的 Amazon Managed Streaming for Apache Kafka 叢集識別碼。如需詳細資訊，請參閱串流擷取。

使用須知

如需使用 Athena 資料目錄時的限制，請參閱 AWS 一般參考中的 Athena 限制。

如需使用時的限制 AWS Glue Data Catalog，請參閱中的「AWS Glue 限制」 AWS 一般參考。

這些限制不適用於 Hive 中繼存放區。

每個資料庫最多 9,900 個結構描述。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的配額和限制。

如要取消註冊結構描述，請使用 DROP SCHEMA 命令。

如要檢視外部結構描述的詳細資訊，請查詢下列系統檢視：

- SVV_EXTERNAL_SCHEMAS
- SVV_EXTERNAL_TABLES
- SVV_EXTERNAL_COLUMNS

範例

以下範例使用美國西部 (奧勒岡) 區域中資料目錄中名為 sampledb 的資料庫建立外部結構描述。將此範例與 Athena 或 AWS Glue 資料目錄搭配使用。

```
create external schema spectrum_schema
```

```
from data catalog
database 'sampledb'
region 'us-west-2'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole';
```

以下範例會建立外部結構描述，並建立名為 spectrum_db 的新外部資料庫。

```
create external schema spectrum_schema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
create external database if not exists;
```

以下範例使用名為 hive_db 的 Hive 中繼存放區資料庫建立外部結構描述。

```
create external schema hive_schema
from hive metastore
database 'hive_db'
uri '172.10.10.10' port 99
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole';
```

以下範例會鏈結角色，以使用 myS3Role 角色存取 Amazon S3，並使用 myAthenaRole 存取資料目錄。如需詳細資訊，請參閱 在 Amazon Redshift Spectrum 中鏈結 IAM 角色。

```
create external schema spectrum_schema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myRedshiftRole,arn:aws:iam::123456789012:role/
myS3Role'
catalog_role 'arn:aws:iam::123456789012:role/myAthenaRole'
create external database if not exists;
```

以下範例會建立外部結構描述，參考 Aurora PostgreSQL 資料庫。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] myRedshiftSchema
FROM POSTGRES
DATABASE 'my_aurora_db' SCHEMA 'my_aurora_schema'
URI 'endpoint to aurora hostname' PORT 5432
IAM_ROLE 'arn:aws:iam::123456789012:role/MyAuroraRole'
SECRET_ARN 'arn:aws:secretsmanager:us-east-2:123456789012:secret:development/
MyTestDatabase-AbCdEf'
```

下列範例會建立外部結構描述，以參照在取用者叢集上匯入的 sales_db。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA 'public';
```

以下範例會建立外部結構描述，參考 Aurora MySQL 資料庫。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] myRedshiftSchema
FROM MYSQL
DATABASE 'my_aurora_db'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyAuroraRole'
SECRET_ARN 'arn:aws:secretsmanager:us-east-2:123456789012:secret:development/
MyTestDatabase-AbCdEf'
```

CREATE EXTERNAL TABLE

在指定的結構描述中建立新的外部資料表。所有外部資料表都必須經由外部結構描述建立。外部結構描述和外部資料表不支援搜尋路徑。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

除了使用「建立外部表」命令建立的外部資料表之外，Amazon Redshift 還可以參考 AWS Lake Formation 目錄 AWS Glue 或 Apache Hive 中繼存放區中定義的外部資料表。使用 CREATE EXTERNAL SCHEMA 命令註冊外部目錄中定義的外部資料庫，並且在 Amazon Redshift 中將外部資料表提供使用。如果外部表存在於 AWS Glue 或 AWS Lake Formation 目錄或 Hive 中繼存儲中，則不需要使用創建外部表創建表。若要檢視外部資料表，請查詢 SVV_EXTERNAL_TABLES 系統畫面。

透過執行 CREATE EXTERNAL TABLE AS 命令，您可以建立依據從查詢欄定義的外部資料表，並將該查詢的結果寫入 Amazon S3。結果為 Apache Parquet 或分隔文字格式。如果外部資料表有一個或多個分割區索引鍵，Amazon Redshift 會根據這些分割區索引鍵來分割新檔案，並自動將新的分割區註冊到外部類別目錄中。如需 CREATE EXTERNAL TABLE AS 的相關資訊，請參閱 使用須知。

您可以用與其他 Amazon Redshift 資料表一起使用的相同 SELECT 語法來查詢外部資料表。您也可以使用 INSERT 語法將新檔案寫入 Amazon S3 上外部資料表的位置。如需詳細資訊，請參閱 INSERT (外部資料表)。

若要建立外部資料表的檢視，請在 CREATE VIEW 陳述式中包含 WITH NO SCHEMA BINDING 子句。

您無法在交易內 (BEGIN … END) 執行 CREATE EXTERNAL TABLE。如需交易的相關資訊，請參閱 可序列化隔離。

所需權限

您必須為外部結構描述的擁有者或超級使用者，始可建立外部資料表。若要轉移外部結構描述的所有權，請使用 ALTER SCHEMA 來變更擁有者。外部資料表的存取權是由外部結構描述的存取權所控制。您無法對外部資料表上的許可執行 GRANT 或 REVOKE。但可改為在外部結構描述授予和撤銷 USAGE。

使用須知 具有有關外部資料表特定權限的其他資訊。

語法

```
CREATE EXTERNAL TABLE
external_schema.table_name
(column_name data_type [, …] )
[ PARTITIONED BY (col_name data_type [, … ] )]
[ { ROW FORMAT DELIMITED row_format |
  ROW FORMAT SERDE 'serde_name'
  [ WITH SERDEPROPERTIES ( 'property_name' = 'property_value' [, ...] ) ] } ]
STORED AS file_format
LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file' }
[ TABLE PROPERTIES ( 'property_name'='property_value' [, ...] ) ]
```

以下是 CREATE EXTERNAL TABLE AS 的語法。

```
CREATE EXTERNAL TABLE
external_schema.table_name
[ PARTITIONED BY (col_name [, … ] ) ]
[ ROW FORMAT DELIMITED row_format ]
STORED AS file_format
LOCATION { 's3://bucket/folder/' }
[ TABLE PROPERTIES ( 'property_name'='property_value' [, ...] ) ]
 AS
 { select_statement }
```

參數

external_schema.table_name

要建立的資料表名稱，以外部結構描述名稱限定。外部資料表必須建立在外部結構描述中。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

資料表名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 多位元組字元，最長可達 4 個位元組。Amazon Redshift 會強制執行每個叢集 9,900 個資料表的限制，包括使用者定義的臨時資料表，以及 Amazon Redshift 在查詢處理或系統維護期間建立的臨時資料表。您也可以選擇使用資料庫名稱來限定資料表名稱。在以下範例中，資料庫名稱為 spectrum_db，外部結構描述名稱為 spectrum_schema，而資料表名稱為 test。

```
create external table spectrum_db.spectrum_schema.test (c1 int)
stored as parquet
location 's3://mybucket/myfolder/';
```

如果指定的資料庫或結構描述不存在，則不會建立資料表，而且陳述式會傳回錯誤。您無法在系統資料庫 template0、template1、padb_harvest 和 sys:internal 中建立資料表或檢視。

資料表名稱對於指定的結構描述來說必須是唯一的。

如需有效名稱的相關資訊，請參閱 名稱與識別碼。

(column_name data_type)

要建立的每個資料欄的名稱和資料類型。

資料欄名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 多位元組字元，最長可達 4 個位元組。您無法指定資料欄名稱 "$path" 或 "$size"。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

Amazon Redshift 預設會以虛擬資料欄 $path 和 $size 建立外部資料表。您可以藉由將 spectrum_enable_pseudo_columns 組態參數設定為 false，以停用工作階段的虛擬資料欄建立。如需詳細資訊，請參閱 虛擬資料欄 。

若已啟用虛擬資料欄，則單一資料表中可定義的資料欄數目上限為 1,598 個。若未啟用虛擬資料欄，則單一資料表中可定義的資料欄數目上限為 1,600 個。

若您要建立「寬資料表」，則務必確定在載入和查詢處理期間，您的資料欄清單未超過中繼結果的資料列寬度界限。如需詳細資訊，請參閱 使用須知。

針對 CREATE EXTERNAL TABLE AS 命令，您不需要欄位清單，因為欄是從查詢中衍生的。

data_type

支援以下 資料類型：

- SMALLINT (INT2)
- INTEGER (INT、INT4)

- BIGINT (INT8)
- DECIMAL (NUMERIC)
- REAL (FLOAT4)
- DOUBLE PRECISION (FLOAT8)
- BOOLEAN (BOOL)
- CHAR (CHARACTER)
- VARCHAR (CHARACTER VARYING)
- VARBYTE (CHARACTER VARYING) – 可以與 Parquet 和 ORC 資料檔案一起使用，並且只能與未分割資料表一起使用。
- DATE – 只可搭配文字、Parquet 或 ORC 資料檔案使用，或做為分割區資料欄使用。
- TIMESTAMP

對於 DATE，您可以使用如下所述的格式。使用數字表示的月份值支援下列格式：

- `mm-dd-yyyy` 例如：`05-01-2017`。此為預設值。
- `yyyy-mm-dd`，其中年份由 2 個以上的數字表示。例如 `2017-05-01`。

使用三個字母縮寫表示的月份值支援下列格式：

- `mmm-dd-yyyy` 例如：`may-01-2017`。此為預設值。
- `dd-mmm-yyyy`，其中年份由 2 個以上的數字表示。例如 `01-may-2017`。
- `yyyy-mmm-dd`，其中年份由 2 個以上的數字表示。例如 `2017-may-01`。

對於始終小於 100 的年份值，年份的計算方式如下：

- 如果年份小於 70，則該年份的計算方式為年份加上 2000。例如，使用 `mm-dd-yyyy` 格式的日期 `05-01-17` 會轉換成 `05-01-2017`。
- 如果年份小於 100 且大於 69，則該年份的計算方式為年份加上 1900。例如，使用 `mm-dd-yyyy` 格式的日期 `05-01-89` 會轉換成 `05-01-1989`。
- 對於以兩個數字表示的年份值，請在前面加上零以使用 4 個數字表示年份。

文字檔案中的時間戳記值格式必須為 `yyyy-mm-dd HH:mm:ss.SSSSSS`，如以下的時間戳記值所示：`2017-05-01 11:30:59.000000`。

VARCHAR 資料欄的長度是以字元組而非字元來定義。例如，VARCHAR(12) 資料欄可包含 12 個單位元組的字元或 6 個 2 位元組的字元。查詢外部資料表時，結果會截斷以配合定義的資料欄大小，而不會傳回錯誤。如需詳細資訊，請參閱 儲存與範圍。

為獲得最佳效能，建議您指定可配合您的資料的最小資料欄大小。若要尋找資料欄中值位元組的大小上限，請使用 OCTET_LENGTH 函數。下列範例會傳回電子郵件資料欄中值的大小上限。

```
select max(octet_length(email)) from users;

max
---
 62
```

PARTITIONED BY (col_name data_type [, …])

此子句會定義包含一個或多個分割區資料欄的分割資料表。每種指定的組合都會另外使用一個資料目錄，這樣可在某些情況下改善查詢效能。資料表資料內並未包含分割資料欄。如果您對 col_name 使用的值與資料表資料欄相同，則會發生錯誤。

建立分割資料表後，使用 ALTER TABLE … ADD PARTITION 陳述式以將新增分割區註冊到外部目錄。當您新增分割區時，會定義 Amazon S3 上包含分割區資料的子資料夾位置。

例如，若資料表 `spectrum.lineitem_part` 是以 `PARTITIONED BY (l_shipdate date)` 定義，執行下列 ALTER TABLE 命令來新增分割區。

```
ALTER TABLE spectrum.lineitem_part ADD PARTITION (l_shipdate='1992-01-29')
LOCATION 's3://spectrum-public/lineitem_partition/l_shipdate=1992-01-29';
```

如果您使用 CREATE EXTERNAL TABLE AS，則不需要執行 ALTER TABLE...ADD PARTITION。Amazon Redshift 會自動在外部目錄中註冊新的分割區。Amazon Redshift 也會根據資料表中定義的一個或多個分割區索引鍵，自動將對應的資料寫入 Amazon S3 中的分割區。

若要檢視分割區，請查詢 SVV_EXTERNAL_PARTITIONS 系統畫面。

> **Note**
>
> 針對 CREATE EXTERNAL TABLE AS 命令，您不需要指定分割區欄位的資料類型，因為此欄位是從查詢衍生的。

ROW FORMAT DELIMITED rowformat

此子句會指定基礎資料的格式。rowformat 可能的值如下：

- LINES TERMINATED BY 'delimiter'
- FIELDS TERMINATED BY 'delimiter'

指定單一 ASCII 字元做為 'delimiter'。您可以使用八進位指定非印刷 ASCII 字元，格式為 '*ddd*'，其中 *d* 是八進位數字 (0 – 7)，最大為 '\177'。以下範例使用八進位指定 BEL (鐘形) 字元。

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\007'
```

如果省略 ROW FORMAT，預設格式會是 DELIMITED FIELDS TERMINATED BY '\A' (標題開頭) 和 LINES TERMINATED BY '\n' (換行符號)。

ROW FORMAT SERDE 'serde_name', [WITH SERDEPROPERTIES ('property_name' = 'property_value' [, ...])]

此子句會指定基礎資料的 SERDE 格式。

'serde_name'

SerDe 的名稱。您可以指定下列格式：

- 或者. 阿帕奇. 哈多普. RegexSerDe
- com. 亞馬遜. 葡萄糖. GrokSerDe
- org.apache.hadoop.hive.serde2.OpenCSVSerde

  這個參數支援下列 SerDe 屬性：

  ```
  'wholeFile' = 'true'
  ```

  將 `wholeFile` 屬性設定為 `true`，以正確剖析 OpenCSV 請求中引號字串內的新行字元 (\n)。
- 組織. 開發. 數據. JsonSer德
  - JSON SERDE 也支援 Ion 檔案。
  - JSON 必須格式正確。
  - 採用 Ion 和 JSON 的時間戳記必須使用 ISO8601 格式。
  - 此參數支援下列 SerDe 屬性 JsonSerDe：

    ```
    'strip.outer.array'='true'
    ```

處理包含含括在外部方括弧 ([...]) 中的一個非常大型陣列的 Ion/JSON 檔案，就好像在陣列內包含多個 JSON 記錄。

- am. 亞馬遜。離子組織. IonHiveSerDe

  除了資料類型外，Amazon ION 格式還提供文字和二進位格式。對於參考 ION 格式資料的外部資料表，您可以將外部資料表中的每一欄對應至 ION 格式資料中的對應元素。如需詳細資訊，請參閱 Amazon Ion。您還需要指定輸入和輸出格式。

WITH SERDEPROPERTIES ('property_name' = 'property_value' [, ...])]

選擇性地指定屬性名稱和值，並以逗號分隔。

如果省略 ROW FORMAT，預設格式會是 DELIMITED FIELDS TERMINATED BY '\A' (標題開頭) 和 LINES TERMINATED BY '\n' (換行符號)。

STORED AS file_format

資料檔案的檔案格式。

有效格式如下：

- PARQUET
- RCFILE (ColumnarSerDe 僅適用於資料使用，不適用) LazyBinaryColumnarSerDe
- SEQUENCEFILE
- TEXTFILE (適用於文字檔案，包括 JSON 檔案)。
- ORC
- AVRO
- INPUTFORMAT 'input_format_classname' OUTPUTFORMAT 'output_format_classname'

CREATE EXTERNAL TABLE AS 命令只支援兩種檔案格式，TEXTFILE 和 PARQUET。

若是 INPUTFORMAT 和 OUTPUTFORMAT，請指定類別名稱，如以下範例所示。

```
'org.apache.hadoop.mapred.TextInputFormat'
```

LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file'}

包含資料檔案的 Amazon S3 儲存貯體或資料夾路徑，或包含 Amazon S3 物件路徑清單的資訊清單檔案。儲存貯體必須與 Amazon 紅移叢集位於相同的 AWS 區域。如需支援的 AWS 區域清單，請參閱Amazon Redshift Spectrum 考量事項。

如果路徑指定的是儲存貯體或資料夾，例如 `'s3://mybucket/custdata/'`，則 Redshift Spectrum 會掃描指定儲存貯體或資料夾以及任何子資料夾裡的檔案。Redshift Spectrum 會忽略隱藏檔案以及開頭為句號或底線的檔案。

如果路徑指定的是資訊清單檔案，則 `'s3://bucket/manifest_file'` 引數必須明確參考單一檔案，例如 `'s3://mybucket/manifest.txt'`。無法參考金鑰前綴。

資訊清單是 JSON 格式的文字檔案，其中列出要從 Amazon S3 載入之每個檔案的 URL 以及檔案的大小 (單位為位元組)。URL 包含檔案的儲存貯體名稱和完整物件路徑。資訊清單中指定的檔案可以位於不同的值區中，但所有值區都必須與 Amazon Redshift 叢集位於相同的 AWS 區域中。如果某個檔案列出兩次，則該檔案會載入兩次。下列範例顯示資訊清單的 JSON，此資訊清單會載入三個檔案。

```
{
  "entries": [
    {"url":"s3://mybucket-alpha/custdata.1", "meta": { "content_length":
 5956875 } },
    {"url":"s3://mybucket-alpha/custdata.2", "meta": { "content_length":
 5997091 } },
    {"url":"s3://mybucket-beta/custdata.1", "meta": { "content_length": 5978675 } }
  ]
}
```

您可以讓包含特定檔案變為強制性。若要這麼做，請在資訊清單中的檔案層級包含 `mandatory` 選項。當您查詢遺失強制性檔案的外部資料表時，SELECT 陳述式會失敗。確定外部資料表定義中包含的所有檔案均存在。如果並非全部存在，則會出現錯誤，顯示找不到第一個強制性檔案。下列範例顯示資訊清單的 JSON，其 `mandatory` 設定為 `true`。

```
{
  "entries": [
    {"url":"s3://mybucket-alpha/custdata.1", "mandatory":true, "meta":
 { "content_length": 5956875 } },
    {"url":"s3://mybucket-alpha/custdata.2", "mandatory":false, "meta":
 { "content_length": 5997091 } },
    {"url":"s3://mybucket-beta/custdata.1", "meta": { "content_length": 5978675 } }
  ]
}
```

若要參考使用 UNLOAD 建立的檔案，您可以使用以 UNLOAD 搭配 MANIFEST 參數所建立的資訊清單。資訊清單檔案與 從 Amazon S3 進行 COPY 的資訊清單檔案相容，但使用不同的索引鍵。未使用的索引鍵會加以忽略。

TABLE PROPERTIES ('property_name'='property_value' [, ...])

此子句會設定資料表屬性的資料表定義。

> **Note**
>
> 資料表屬性區分大小寫。

'compression_type'='value'

如果檔案名稱未包含副檔名，此屬性會設定要使用的壓縮類型。如果您設定此屬性，但是有副檔名，則會忽略副檔名並使用屬性所設定的值。壓縮類型的有效值如下所示：

- bzip2
- gzip
- 無
- snappy

'data_cleansing_enabled'='true / false’

此屬性會設定資料表的資料處理是否開啟。當 'data_cleansing_enabled' 設定為 true 時，資料表的資料處理為開啟狀態。當 'data_cleansing_enabled' 設定為 false 時，資料表的資料處理為關閉狀態。以下是此屬性所控制的資料表層級資料處理屬性清單：

- column_count_mismatch_handling
- invalid_char_handling
- numeric_overflow_handling
- replacement_char
- surplus_char_handling

如需範例，請參閱 資料處理範例。

'invalid_char_handling'='value'

指定當查詢結果包含無效的 UTF-8 字元值時要執行的動作。您可以指定下列動作：

DISABLED

不執行無效的字元處理。

FAIL

取消傳回的資料包含無效 UTF-8 值的查詢。

SET_TO_NULL

以 null 取代無效的 UTF-8 值。

DROP_ROW

將資料列中的每個值取代為 null。

REPLACE

以您使用 `replacement_char` 指定的取代字元取代無效字元。

'replacement_char'='character’

指定當您將 `invalid_char_handling` 設定為 `REPLACE` 時要使用的取代字元。

'numeric_overflow_handling'='value’

指定當 ORC 資料包含大於資料欄定義 (例如 SMALLINT 或 int16) 的整數 (例如，BIGINT 或 int64) 時，要執行的動作。您可以指定下列動作：

DISABLED

無效字元處理已關閉。

FAIL

當資料包含無效字元時，取消查詢。

SET_TO_NULL

將無效字符設定為 null。

DROP_ROW

將資料列中的每個值設定為 null。

'surplus_bytes_handling'='value'

針對包含 VARBYTE 資料的資料欄，指定當載入的資料超過所定義的資料類型長度時應如何處理。根據預設，Redshift Spectrum 會針對超出欄寬度的資料，將值設定為 null。

您可以指定當查詢傳回超過資料類型長度的資料時，執行下列動作：

SET_TO_NULL

以 null 取代超過欄寬的資料。

DISABLED

不執行多餘位元組處理。

FAIL

取消傳回資料超出欄寬的查詢。

DROP_ROW

捨棄包含超出欄寬之資料的所有資料列。

TRUNCATE

如果字元超過為欄定義的字元數目上限，則移除字元。

'surplus_char_handling'='value'

針對包含 VARCHAR、CHAR 或字串資料的資料欄，指定當載入的資料超過所定義的資料類型長度時應如何處理。根據預設，Redshift Spectrum 會針對超出欄寬度的資料，將值設定為 null。

您可以指定當查詢傳回超過欄寬的資料時，執行下列動作：

SET_TO_NULL

以 null 取代超過欄寬的資料。

DISABLED

不執行多餘字元處理。

FAIL

取消傳回資料超出欄寬的查詢。

DROP_ROW

將資料列中的每個值取代為 null。

TRUNCATE

如果字元超過為欄定義的字元數目上限，則移除字元。

'column_count_mismatch_handling'='value’

識別檔案包含的資料列值是否少於或多於外部資料表定義中指定的欄數。此屬性僅適用於未壓縮的文字檔案格式。您可以指定下列動作：

DISABLED

欄計數不相符處理已關閉。

FAIL

如果偵測到資料欄計數不相符，則查詢失敗。

SET_TO_NULL

使用 NULL 填入遺漏值，並忽略每一列中的其他值。

DROP_ROW

從掃描中捨棄包含欄計數不相符錯誤的所有資料列。

'numRows'='row_count'

此屬性會設定資料表定義的 numRows 值。若要明確更新外部資料表的統計資料，請設定 numRows 屬性以指出資料表的大小。Amazon Redshift 不會分析外部資料表來產生查詢最佳化工具用來產生查詢計劃的資料表統計資料。如果資料表統計資訊沒有為外部資料表進行設定，則 Amazon Redshift 會以「外部資料表較大而本機資料表較小」的假設來產生查詢執行計畫。

'skip.header.line.count'='line_count'

此屬性會設定每個來源檔案開頭要略過的資料列數。

'serialization.null.format'=' '

此屬性會指定，欄位中提供的文字有完全相符項目時，Spectrum 應傳回 NULL 值。

'orc.schema.resolution'='mapping_type'

此屬性會為使用 ORC 日期格式的資料表設定資料欄映射類型。將針對其他所有日期格式忽略此屬性。

資料欄映射類型的有效值如下所示：

- name
- position

如果忽略 orc.schema.resolution 屬性，則預設會依名稱映射資料欄。如果 orc.schema.resolution 設為 'name' 或 'position' 以外的任何值，則會依位置映射資料欄。如需資料欄映射的相關資訊，請參閱 將外部資料表資料欄映射到 ORC 資料欄。

> **Note**
>
> COPY 命令只會依位置映射至 ORC 資料檔案。orc.schema.resolution 資料表屬性對於 COPY 命令行為沒有影響。

'write.parallel'='on/off'

設定是否將 CREATE EXTERNAL TABLE AS 平行寫入資料的屬性。依預設，CREATE EXTERNAL TABLE AS 會根據叢集中的分割數，將資料平行寫入多個檔案。預設選項為開啟。當 'write.parallel' 設為關閉時，CREATE EXTERNAL TABLE AS 會連續寫入一個或多個資料檔案到 Amazon S3。此資料表屬性也適用於相同外部資料表的任何後續 INSERT 陳述式。

'write.maxfilesize.mb'='size'

設定由 CREATE EXTERNAL TABLE AS 寫入 Amazon S3 的每個檔案之大小上限 (以 MB 為單位) 屬性。大小必須是介於 5 到 6200 之間的有效整數。預設的檔案大小上限為 6,200 MB。此資料表屬性也適用於相同外部資料表的任何後續 INSERT 陳述式。

'write.kms.key.id'='value'

您可以指定 AWS Key Management Service 金鑰來為 Amazon S3 物件啟用伺服器端加密 (SSE)，其中值為下列其中一項：

- `auto`使用存放在 Amazon S3 儲存貯體中的預設 AWS KMS 金鑰。
- 您指定用來加密資料的 kms-key。

select_statement

透過定義任何查詢，將一或多個列插入外部資料表的陳述式。查詢產生的所有列都會根據表格定義，以文字或 Parquet 格式寫入到 Amazon S3。

範例

您可在 範例 取得範例集合。

使用須知

本主題包含 CREATE EXTERNAL TABLE 的使用須知。您無法使用與標準 Amazon Redshift 資料表相同的資源檢視 Amazon Redshift Spectrum 資料表的詳細資訊，例如 PG_TABLE_DEF、STV_TBL_PERM、PG_CLASS 或 information_schema。如果您的商

業智慧或分析工具無法識別 Redshift Spectrum 外部資料表，請將您的應用程式設定為查詢 SVV_EXTERNAL_TABLES 與 SVV_EXTERNAL_COLUMNS。

CREATE EXTERNAL TABLE AS

在某些情況下，您可能會在 AWS Glue 資料目錄、外部目錄或 Apache Hive 中繼存放區上執行「建立 AWS Lake Formation 外部表格 AS」命令。在這種情況下，您可以使用 AWS Identity and Access Management (IAM) 角色來建立外部結構描述。此 IAM 角色必須同時具有 Amazon S3 的讀取和寫入許可。

如果您使用 Lake Formation 目錄，則 IAM 角色必須具有在目錄中建立資料表的許可。在此案例中，它也必須具有目標 Amazon S3 路徑上的資料湖位置許可。此 IAM 角色會成為新 AWS Lake Formation 資料表的擁有者。

為了確保檔案名稱是唯一的，Amazon Redshift 依預設會針對每個上傳到 Amazon S3 的檔案名稱使用下列格式。

*<date>*_*<time>*_*<microseconds>*_*<query_id>*_*<slice-number>*_part_*<part-number>*.*<format>*.

例如，20200303_004509_810669_1007_0001_part_00.parquet。

執行 CREATE EXTERNAL TABLE AS 命令時，請考慮以下幾點：

- Amazon S3 位置必須為空。
- 使用 STORED AS 子句時，Amazon Redshift 僅支援 PARQUET 和 TEXTFILE 格式。
- 您不需要定義欄位定義清單。新外部資料表的欄位名稱和欄位資料類型是直接從 SELECT 查詢衍生。
- 您不需要在 PARTITIONED BY 子句中定義分割區欄位的資料類型。如果您指定分割區索引鍵，此欄位的名稱必須存在於 SELECT 查詢結果中。當有多個分割資料欄時，其在 SELECT 查詢中的順序並不重要。Amazon Redshift 會使用其在 PARTITIONED BY 子句中定義的順序來建立外部資料表。
- Amazon Redshift 會根據分割區索引鍵值，自動將輸出檔案分割到分割區資料夾中。依預設，Amazon Redshift 會從輸出檔案中移除分割區欄位。
- 不支援 LINES TERMINATED BY 'delimiter' 子句。
- 不支援 ROW FORMAT SERDE 'serde_name' 子句。
- 不支援使用資訊清單檔案。因此，您無法將 LOCATION 子句定義為 Amazon S3 上的資訊清單檔案。
- Amazon Redshift 會自動更新命令末尾的 'numRows' 資料表屬性。

- 'compression_type' 資料表屬性只接受 'none' 或 'snappy' 的 PARQUET 檔案格式。
- Amazon Redshift 不允許外部 SELECT 查詢中的 LIMIT 子句。相反的，您可以使用巢狀 LIMIT 子句。
- 您可以使用 STL_UNLOAD_LOG 來追蹤由每個 CREATE EXTERNAL TABLE AS 操作寫入到 Amazon S3 的檔案。

建立和查詢外部資料表的許可

若要建立外部資料表，請確定您是外部結構描述或超級使用者的擁有者。若要轉移外部結構描述的所有權，請使用 ALTER SCHEMA。下列範例會將 spectrum_schema 結構描述的擁有者變更為 newowner。

```
alter schema spectrum_schema owner to newowner;
```

若要執行 Redshift Spectrum 查詢，您需要以下許可：

- 結構描述使用許可
- 在目前資料庫建立暫時資料表的許可

下列範例可在結構描述 spectrum_schema 上授予使用許可至 spectrumusers 使用者群組。

```
grant usage on schema spectrum_schema to group spectrumusers;
```

下列範例可在資料庫 spectrumdb 上授予臨時許可至 spectrumusers 使用者群組。

```
grant temp on database spectrumdb to group spectrumusers;
```

虛擬資料欄

Amazon Redshift 預設會以虛擬資料欄 $path 和 $size 建立外部資料表。選擇這些欄位以檢視 Amazon S3 上資料檔案的路徑，以及由查詢傳回的每列資料檔案大小。$path 與 $size 欄位名稱必須以雙引號分隔。SELECT * 子句不會傳回虛擬資料欄。您必須在查詢中明確包含 $path 和 $size 欄位名稱，如以下範例所示。

```
select "$path", "$size"
from spectrum.sales_part
where saledate = '2008-12-01';
```

您可以藉由將 spectrum_enable_pseudo_columns 組態參數設定為 false，以停用工作階段的虛擬資料欄建立。

> **Important**
>
> 選擇 $size 或 $path 會產生費用，因為 Redshift Spectrum 會掃描 Amazon S3 中的資料檔案以判斷結果集的大小。如需詳細資訊，請參閱 Amazon Redshift 定價。

設定資料處理選項

您可以設定資料表參數，為要在外部資料表中查詢的資料指定輸入處理，包括：

- 包含 VARCHAR，CHAR 和字串資料之資料欄中的多餘字元。如需詳細資訊，請參閱外部資料表屬性 `surplus_char_handling`。
- 包含 VARCHAR，CHAR 和字串資料之資料欄中的無效字元。如需詳細資訊，請參閱外部資料表屬性 `invalid_char_handling`。
- 當您為外部資料表屬性 `invalid_char_handling` 指定 REPLACE 時要使用的取代字元。
- 在包含整數和十進位資料的資料欄中進行轉換溢位處理。如需詳細資訊，請參閱外部資料表屬性 `numeric_overflow_handling`。
- 在包含 VARBYTE 資料的資料欄中，Surplus_bytes_handling 可指定多餘位元組的輸入處理。如需詳細資訊，請參閱外部資料表屬性 `surplus_bytes_handling`。

範例

以下範例在名為 `spectrum` 的 Amazon Redshift 外部結構描述中建立名為 SALES 的資料表。該資料位於 Tab 鍵分隔的文字檔案中。TABLE PROPERTIES 子句會將 numRows 屬性設定為 170,000 個資料列。

視您用來執行 CREATE EXTERNAL TABLE 的身分而定，您可能需要設定 IAM 許可。最佳做法是，建議您將許可政策附加到 IAM 角色，然後根據需要將其指派給使用者和群組。如需詳細資訊，請參閱 Amazon Redshift 中的身分和存取管理。

```
create external table spectrum.sales(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
```

```
saledate date,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales/'
table properties ('numRows'='170000');
```

下列範例會建立使用 JsonSerDe 來參照 JSON 格式資料的資料表。

```
create external table spectrum.cloudtrail_json (
event_version int,
event_id bigint,
event_time timestamp,
event_type varchar(10),
awsregion varchar(20),
event_name varchar(max),
event_source varchar(max),
requesttime timestamp,
useragent varchar(max),
recipientaccountid bigint)
row format serde 'org.openx.data.jsonserde.JsonSerDe'
with serdeproperties (
'dots.in.keys' = 'true',
'mapping.requesttime' = 'requesttimestamp'
) location 's3://mybucket/json/cloudtrail';
```

下列 CREATE EXTERNAL TABLE AS 範例會建立未分割的外部資料表。然後它會寫入 SELECT 查詢結果，作為 Apache Parquet 到目標 Amazon S3 位置。

```
CREATE EXTERNAL TABLE spectrum.lineitem
STORED AS parquet
LOCATION 'S3://mybucket/cetas/lineitem/'
AS SELECT * FROM local_lineitem;
```

下列範例會建立已分割的外部資料表，並在 SELECT 查詢中包含分割區欄位。

```
CREATE EXTERNAL TABLE spectrum.partitioned_lineitem
PARTITIONED BY (l_shipdate, l_shipmode)
```

```
STORED AS parquet
LOCATION 'S3://mybucket/cetas/partitioned_lineitem/'
AS SELECT l_orderkey, l_shipmode, l_shipdate, l_partkey FROM local_table;
```

如需外部資料目錄中現有資料庫的清單，請查詢 SVV_EXTERNAL_DATABASES 系統畫面。

```
select eskind,databasename,esoptions from svv_external_databases order by databasename;
```

```
eskind | databasename | esoptions
-------+--------------
+-------------------------------------------------------------------------------
     1 | default      | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
     1 | sampledb     | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
     1 | spectrumdb   | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

若要檢視外部資料表的詳細資訊，請查詢 SVV_EXTERNAL_TABLES 和 SVV_EXTERNAL_COLUMNS 系統畫面。

以下範例會查詢 SVV_EXTERNAL_TABLES 檢視。

```
select schemaname, tablename, location from svv_external_tables;
```

```
schemaname | tablename            | location
-----------+----------------------
+-----------------------------------------------------
spectrum   | sales                | s3://redshift-downloads/tickit/spectrum/sales
spectrum   | sales_part           | s3://redshift-downloads/tickit/spectrum/
sales_partition
```

以下範例會查詢 SVV_EXTERNAL_COLUMNS 檢視。

```
select * from svv_external_columns where schemaname like 'spectrum%' and tablename
 ='sales';
```

```
schemaname | tablename | columnname | external_type | columnnum | part_key
-----------+-----------+------------+---------------+-----------+---------
spectrum   | sales     | salesid    | int           |         1 |        0
spectrum   | sales     | listid     | int           |         2 |        0
```

```
spectrum   | sales     | sellerid   | int           |         3 |        0
spectrum   | sales     | buyerid    | int           |         4 |        0
spectrum   | sales     | eventid    | int           |         5 |        0
spectrum   | sales     | saledate   | date          |         6 |        0
spectrum   | sales     | qtysold    | smallint      |         7 |        0
spectrum   | sales     | pricepaid  | decimal(8,2)  |         8 |        0
spectrum   | sales     | commission | decimal(8,2)  |         9 |        0
spectrum   | sales     | saletime   | timestamp     |        10 |        0
```

若要檢視資料表分割區，請使用下列查詢。

```
select schemaname, tablename, values, location
from svv_external_partitions
where tablename = 'sales_part';
```

```
schemaname | tablename  | values         | location
-----------+------------+----------------
+-------------------------------------------------------------------------
spectrum   | sales_part | ["2008-01-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-01
spectrum   | sales_part | ["2008-02-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-02
spectrum   | sales_part | ["2008-03-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-03
spectrum   | sales_part | ["2008-04-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-04
spectrum   | sales_part | ["2008-05-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-05
spectrum   | sales_part | ["2008-06-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-06
spectrum   | sales_part | ["2008-07-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-07
spectrum   | sales_part | ["2008-08-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-08
spectrum   | sales_part | ["2008-09-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-09
spectrum   | sales_part | ["2008-10-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-10
spectrum   | sales_part | ["2008-11-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-11
spectrum   | sales_part | ["2008-12-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-12
```

下列範例會傳回外部資料表相關的資料檔案大小總和。

```
select distinct "$path", "$size"
   from spectrum.sales_part;

 $path                                                                      | $size
----------------------------------------------------------------------------+-------
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/ |  1616
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/ |  1444
s3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/ |  1444
```

資料分割範例

若要建立以日期分割的外部資料表，請執行以下命令。

```
create external table spectrum.sales_part(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
partitioned by (saledate date)
row format delimited
fields terminated by '|'
stored as textfile
location 's3://redshift-downloads/tickit/spectrum/sales_partition/'
table properties ('numRows'='170000');
```

請執行下列 ALTER TABLE 命令以新增分割區。

```
alter table spectrum.sales_part
add if not exists partition (saledate='2008-01-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-01/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-02-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-02/';
alter table spectrum.sales_part
```

```
add if not exists partition (saledate='2008-03-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-03/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-04-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-04/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-05-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-05/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-06-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-06/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-07-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-07/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-08-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-08/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-09-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-09/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-10-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-10/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-11-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-11/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-12-01')
location 's3://redshift-downloads/tickit/spectrum/sales_partition/saledate=2008-12/';
```

若要從分割資料表選取資料，請執行下列查詢。

```
select top 10 spectrum.sales_part.eventid, sum(spectrum.sales_part.pricepaid)
from spectrum.sales_part, event
where spectrum.sales_part.eventid = event.eventid
  and spectrum.sales_part.pricepaid > 30
  and saledate = '2008-12-01'
group by spectrum.sales_part.eventid
order by 2 desc;
```

```
eventid | sum
--------+---------
    914 | 36173.00
```

```
   5478 | 27303.00
   5061 | 26383.00
   4406 | 26252.00
   5324 | 24015.00
   1829 | 23911.00
   3601 | 23616.00
   3665 | 23214.00
   6069 | 22869.00
   5638 | 22551.00
```

若要檢視外部資料表分割區，請查詢 SVV_EXTERNAL_PARTITIONS 系統畫面。

```
select schemaname, tablename, values, location from svv_external_partitions
where tablename = 'sales_part';
```

```
schemaname | tablename  | values         | location
-----------+------------+----------------
+-----------------------------------------------
spectrum   | sales_part | ["2008-01-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-01
spectrum   | sales_part | ["2008-02-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-02
spectrum   | sales_part | ["2008-03-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-03
spectrum   | sales_part | ["2008-04-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-04
spectrum   | sales_part | ["2008-05-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-05
spectrum   | sales_part | ["2008-06-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-06
spectrum   | sales_part | ["2008-07-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-07
spectrum   | sales_part | ["2008-08-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-08
spectrum   | sales_part | ["2008-09-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-09
spectrum   | sales_part | ["2008-10-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-10
spectrum   | sales_part | ["2008-11-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-11
spectrum   | sales_part | ["2008-12-01"] | s3://redshift-downloads/tickit/spectrum/
sales_partition/saledate=2008-12
```

資料列格式範例

下面顯示的範例會針對以 AVRO 格式儲存的資料檔案，指定 ROW FORMAT SERDE 參數。

```
create external table spectrum.sales(salesid int, listid int, sellerid int,
 buyerid int, eventid int, dateid int, qtysold int, pricepaid decimal(8,2), comment
 VARCHAR(255))
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
WITH SERDEPROPERTIES ('avro.schema.literal'='{\"namespace\": \"dory.sample\",\"name\":
 \"dory_avro\",\"type\": \"record\", \"fields\": [{\"name\":\"salesid\", \"type\":\"int
\"},
{\"name\":\"listid\", \"type\":\"int\"},
{\"name\":\"sellerid\", \"type\":\"int\"},
{\"name\":\"buyerid\", \"type\":\"int\"},
{\"name\":\"eventid\",\"type\":\"int\"},
{\"name\":\"dateid\",\"type\":\"int\"},
{\"name\":\"qtysold\",\"type\":\"int\"},
{\"name\":\"pricepaid\", \"type\": {\"type\": \"bytes\", \"logicalType\": \"decimal\",
 \"precision\": 8, \"scale\": 2}}, {\"name\":\"comment\",\"type\":\"string\"}]}')
STORED AS AVRO
location 's3://mybucket/avro/sales' ;
```

下面顯示了使 RegEx用指定行格式 SERDE 參數的示例。

```
create external table spectrum.types(
cbigint bigint,
cbigint_null bigint,
cint int,
cint_null int)
row format serde 'org.apache.hadoop.hive.serde2.RegexSerDe'
with serdeproperties ('input.regex'='([^\\x01]+)\\x01([^\\x01]+)\\x01([^\\x01]+)\
\x01([^\\x01]+)')
stored as textfile
location 's3://mybucket/regex/types';
```

下面顯示的範例會使用 Grok 指定 ROW FORMAT SERDE 參數。

```
create external table spectrum.grok_log(
timestamp varchar(255),
pid varchar(255),
loglevel varchar(255),
progname varchar(255),
```

```
message varchar(255))
row format serde 'com.amazonaws.glue.serde.GrokSerDe'
with serdeproperties ('input.format'='[DFEWI], \\[%{TIMESTAMP_ISO8601:timestamp} #
%{POSINT:pid:int}\\] *(?<loglevel>:DEBUG|FATAL|ERROR|WARN|INFO) -- +%{DATA:progname}:
 %{GREEDYDATA:message}')
stored as textfile
location 's3://mybucket/grok/logs';
```

下面範例會示範在 S3 儲存貯體中定義 Amazon S3 伺服器存取日誌。您可以使用 Redshift Spectrum 查詢 Amazon S3 存取日誌。

```
CREATE EXTERNAL TABLE spectrum.mybucket_s3_logs(
bucketowner varchar(255),
bucket varchar(255),
requestdatetime varchar(2000),
remoteip varchar(255),
requester varchar(255),
requested varchar(255),
operation varchar(255),
key varchar(255),
requesturi_operation varchar(255),
requesturi_key varchar(255),
requesturi_httpprotoversion varchar(255),
httpstatus varchar(255),
errorcode varchar(255),
bytessent bigint,
objectsize bigint,
totaltime varchar(255),
turnaroundtime varchar(255),
referrer varchar(255),
useragent varchar(255),
versionid varchar(255)
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
'input.regex' = '([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 \"([^ ]*)\\s*([^ ]*)\\s*([^ ]*)\" (- |[^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\") ([^ ]*).*$')
LOCATION 's3://mybucket/s3logs’;
```

下面顯示的範例會針對 ION 格式資料，指定 ROW FORMAT SERDE 參數。

```
CREATE EXTERNAL TABLE tbl_name (columns)
```

```
ROW FORMAT SERDE 'com.amazon.ionhiveserde.IonHiveSerDe'
STORED AS
INPUTFORMAT 'com.amazon.ionhiveserde.formats.IonInputFormat'
OUTPUTFORMAT 'com.amazon.ionhiveserde.formats.IonOutputFormat'
LOCATION 's3://s3-bucket/prefix'
```

資料處理範例

下列範例會存取檔案：spi_global_rankings.csv。您可以將 spi_global_rankings.csv 檔案上傳至 Amazon S3 儲存貯體以嘗試下列範例。

以下範例會建立外部結構描述 schema_spectrum_uddh 和資料庫 spectrum_db_uddh。對於aws-account-id，輸入您的 AWS 帳戶 ID，然role-name後輸入您的 Redshift 頻譜角色名稱。

```
create external schema schema_spectrum_uddh
from data catalog
database 'spectrum_db_uddh'
iam_role 'arn:aws:iam::aws-account-id:role/role-name'
create external database if not exists;
```

以下範例會在外部結構描述 schema_spectrum_uddh 中建立外部資料表 soccer_league。

```
CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league
(
  league_rank smallint,
  prev_rank   smallint,
  club_name   varchar(15),
  league_name varchar(20),
  league_off  decimal(6,2),
  league_def  decimal(6,2),
  league_spi  decimal(6,2),
  league_nspi integer
)
ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n\l'
stored as textfile
LOCATION 's3://spectrum-uddh/league/'
table properties ('skip.header.line.count'='1');
```

檢查 soccer_league 資料表中的列數。

```
select count(*) from schema_spectrum_uddh.soccer_league;
```

列數會隨其顯示。

```
count
645
```

下列查詢會顯示前 10 名的俱樂部。因為俱樂部 Barcelona 在字串中有無效的字元，所以會顯示 NULL 做為名稱。

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

```
league_rank club_name league_name   league_nspi
1  Manchester City Barclays Premier Lea  34595
2  Bayern Munich German Bundesliga  34151
3  Liverpool Barclays Premier Lea  33223
4  Chelsea  Barclays Premier Lea  32808
5  Ajax  Dutch Eredivisie  32790
6  Atletico  Madrid Spanish Primera Divi 31517
7  Real Madrid Spanish Primera Divi  31469
8  NULL         Spanish Primera Divi            31321
9  RB Leipzig German Bundesliga  31014
10  Paris Saint-Ger French Ligue 1   30929
```

下列範例會修改 soccer_league 資料表，以指定 invalid_char_handling、replacement_char 和 data_cleansing_enabled 外部資料表屬性來插入問號 (?) 替代非預期字元。

```
alter  table schema_spectrum_uddh.soccer_league
set table properties
 ('invalid_char_handling'='REPLACE','replacement_char'='?','data_cleansing_enabled'='true');
```

下列範例會針對排名 1 到 10 的團隊查詢 soccer_league 資料表。

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

由於資料表屬性已修改，因此結果會顯示前 10 名的俱樂部，並在俱樂部 Barcelona 的第八列包含問號 (?) 替代字元。

```
league_rank club_name league_name  league_nspi
1  Manchester City Barclays Premier Lea 34595
2  Bayern Munich German Bundesliga 34151
3  Liverpool Barclays Premier Lea 33223
4  Chelsea  Barclays Premier Lea 32808
5  Ajax  Dutch Eredivisie 32790
6  Atletico Madrid Spanish Primera Divi 31517
7  Real Madrid Spanish Primera Divi 31469
8  Barcel?na Spanish Primera Divi 31321
9  RB Leipzig German Bundesliga 31014
10  Paris Saint-Ger French Ligue 1  30929
```

下列範例會修改資料表 soccer_league 來指定 invalid_char_handling 外部資料表屬性，以捨棄含有非預期字元的資料列。

```
alter table schema_spectrum_uddh.soccer_league
set table properties
 ('invalid_char_handling'='DROP_ROW','data_cleansing_enabled'='true');
```

下列範例會針對排名 1 到 10 的團隊查詢 soccer_league 資料表。

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

結果會顯示熱門俱樂部，不包括俱樂部 Barcelona 的第八列。

```
league_rank   club_name         league_name            league_nspi
1             Manchester City   Barclays Premier Lea   34595
2             Bayern Munich     German Bundesliga      34151
3             Liverpool         Barclays Premier Lea   33223
4             Chelsea           Barclays Premier Lea   32808
5             Ajax              Dutch Eredivisie       32790
6             Atletico Madrid   Spanish Primera Divi   31517
7             Real Madrid       Spanish Primera Divi   31469
9             RB Leipzig        German Bundesliga      31014
10            Paris Saint-Ger   French Ligue 1         30929
```

CREATE EXTERNAL VIEW (預覽)

這是適用於 Amazon Redshift 的資料目錄中的發行前版本文件檢視，屬於預覽版本。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。 AWS 區域 會列出目前帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的建立叢集。
5. 選擇建立叢集按鈕以建立預覽叢集。

   > **Note**
   >
   > `preview_2023` 軌跡是最近可用的預覽軌跡。此軌跡僅支援使用 RA3 節點類型建立叢集。不支援節點類型 DC2 和任何較舊的節點類型。

6. 當您的預覽叢集可用時，請使用 SQL 用戶端載入和查詢資料。

Data Catalog 視觀表預覽功能僅適用於以下區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)
- 亞太區域 (東京) (ap-northeast-1)

- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

您也可以建立預覽工作群組來測試 Data Catalog 視觀表。您無法在生產環境中使用這些功能，也無法將工作群組移至另一個工作群組。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。如需如何建立預覽工作群組的指示，請參閱建立預覽工作群組。

在 Data Catalog 中建立檢視。Data Catalog 視觀表是一種單一檢視結構描述，可使用其他 SQL 引擎 (例如 Amazon Athena 和 Amazon EMR)。您可以從您選擇的引擎中查詢檢視。如需有關 Data Catalog 視觀表的詳細資訊，請參閱建立 Data Catalog 視觀表 (預覽)。

語法

```
CREATE EXTERNAL VIEW schema_name.view_name [ IF NOT EXISTS ]
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
 external_schema_name.view_name}
AS query_definition;
```

參數

schema_name.view_name

附加到 AWS Glue 資料庫的結構描述，後面接著檢視的名稱。

受保護

指定只有在 query_define 中的查詢可以成功完成時，才應完成 CREATE EXTERNAL VIEW 命令。

IF NOT EXISTS

如果檢視不存在，則建立檢視。

catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name | external_schema_name.view_name

建立視觀表時所使用的結構描述標記法。您可以指定使用 AWS Glue Data Catalog您建立的 Glue 資料庫或您建立的外部結構描述。如需詳細資訊，請參閱CREATE DATABASE 和 CREATE EXTERNAL SCHEMA。

query_definition

Amazon Redshift 執行以變更檢視的 SQL 查詢的定義。

範例

下列範例會建立名為 sample_schema.glue_data_catalog_view 的 Data Catalog 視觀表。

```
CREATE EXTERNAL PROTECTED VIEW sample_schema.glue_data_catalog_view IF NOT EXISTS
AS SELECT * FROM sample_database.remote_table "remote-table-name";
```

CREATE FUNCTION

使用 SQL SELECT 子句或 Python 程式，建立新的純量使用者定義函數 (UDF)。

如需詳細資訊和範例，請參閱 建立使用者定義的函數。

所需權限

您必須具有下列其中一種方式的權限，才能執行「建立或取代」功能：

- 對於 CREATE FUNCTION：
  - 超級使用者可以使用受信任和不受信任的語言來建立函數。
  - 具有 CREATE [OR REPLACE] FUNCTION 權限的使用者可以使用信任語言建立函數。
- 對於 REPLACE FUNCTION：
  - 超級使用者
  - 具有 CREATE [OR REPLACE] FUNCTION 權限的使用者
  - 函數擁有者

語法

```
CREATE [ OR REPLACE ] FUNCTION f_function_name
( { [py_arg_name  py_arg_data_type |
sql_arg_data_type } [ , ... ] ] )
RETURNS data_type
{ VOLATILE | STABLE | IMMUTABLE }
AS $$
  { python_program | SELECT_clause }
$$ LANGUAGE { plpythonu | sql }
```

參數

OR REPLACE

指定已有相同名稱和輸入引數資料類型 (或簽章) 的函數存在時，取代現有函數。您可以將函數取代為定義一組相同資料類型的新函數。您必須是超級使用者才能取代函數。

如果您定義的函數與現有函數同名，但簽章不同，則會建立新函數。換言之，函數名稱將會過載。如需詳細資訊，請參閱 多載函數名稱。

f_function_name

函數的名稱。如果您指定結構描述名稱 (例如 `myschema.myfunction`)，則會使用指定的結構描述建立函數。否則，函數會在目前結構描述中建立。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

我們建議您在所有 UDF 名稱前加上 `f_`。Amazon Redshift 會保留 `f_` 字首，專供 UDF 名稱使用，因此，使用 `f_` 字首，您可以確保 UDF 名稱不會與任何現有或未來的 Amazon Redshift 內建 SQL 函數名稱發生衝突。如需詳細資訊，請參閱 命名 UDF。

若輸入引數的資料類型不同，則您可以定義多個擁有相同函數名稱的函數。換言之，函數名稱將會過載。如需詳細資訊，請參閱 多載函數名稱。

py_arg_name py_arg_data_type | sql_arg_data type

若是 Python UDF，此為輸入引數名稱和資料類型的清單。若是 SQL UDF，此為資料類型的清單，不包含引數名稱。在 Python UDF 中，使用引數名稱來參考引數。在 SQL UDF 中，根據引數清單中的引數順序，使用 $1、$2 等來參考引數。

若是 SQL UDF，輸入和傳回資料類型可以是任何標準 Amazon Redshift 資料類型。使用 Python UDF 時，輸入及傳回的資料類型可為 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE PRECISION、BOOLEAN、CHAR、VARCHAR、DATE 或 TIMESTAMP。此外，Python 使用者定義函數 (UDF) 支援 ANYELEMENT 資料類型。這會根據執行時提供的對應引數資料類型，自動轉換為標準資料類型。如果有多個引數使用 ANYELEMENT，根據清單中的第一個 ANYELEMENT 引數而定，這些引數在執行時都會解析為相同資料類型。如需詳細資訊，請參閱 Python UDF 資料類型 及 資料類型。

您最多可以指定 32 個引數。

RETURNS data_type

函數所傳回值的資料類型。RETURNS 資料類型可以是任何標準 Amazon Redshift 資料類型。此外，Python UDF 還可以使用資料類型 ANYELEMENT，該資料類型會根據執行時提供的引數自動轉換為標準資料類型。如果您指定 ANYELEMENT 做為傳回資料類型，則至少有一個引數必須使用 ANYELEMENT。實際的傳回資料類型將與呼叫函數時提供給 ANYELEMENT 引數的資料類型相同。如需詳細資訊，請參閱 Python UDF 資料類型。

VOLATILE | STABLE | IMMUTABLE

通知查詢最佳化工具有關函數的波動情形。

如果您將函數標示為最嚴格的有效波動類別，將會得到最理想的最佳化結果。不過，如果類別太嚴格，則最佳化工具可能會錯誤地略過某些呼叫，導致產生不正確的結果集。從最低嚴格程度開始，依嚴格程度排列的波動類別如下所示：

- VOLATILE
- STABLE
- IMMUTABLE

VOLATILE

假設引數相同，即使是針對單一陳述式中的資料列，函數也可能在後續呼叫中傳回不同的結果。查詢最佳化工具無法對波動函數的行為做出任何假設，因此，使用波動函數的查詢必須針對每個輸入資料列重新評估函數。

STABLE

假設引數相同，函數一定會針對單一陳述式內處理的所有資料列傳回相同結果。在不同陳述式中呼叫函數時，函數可能傳回不同結果。此類別可讓最佳化工具將單一陳述式中的多次函數呼叫最佳化，成為陳述式的單一呼叫。

IMMUTABLE

假設引數相同，函數一律傳回相同結果，而且永遠不變。當查詢呼叫具有常數引數的 IMMUTABLE 函數時，最佳化工具會預先評估函數。

AS $$ statement $$

包圍要執行之陳述式的結構。常值關鍵字 AS $$ 和 $$ 是必要的。

Amazon Redshift 會要求您使用稱為 $ 符號引用的格式包圍函數中的陳述式。包圍範圍當中的任何內容都會原封不動傳遞。您不需要逸出任何特殊字元，因為字串的內容是逐字撰寫。

使用 $ 符號引用時，您會使用一組 $$ 符號配對表示要執行之陳述式的開頭和結尾，如以下範例所示。

```
$$ my statement $$
```

您也可以選擇在每組配對的兩個 $ 符號之間指定字串來協助識別陳述式。您在包圍配對的開頭和結尾中使用的字串必須相同。此字串區分大小寫，且遵循與未加引號的識別碼相同的限制條件，不過後者不可包含 $ 符號。下列範例使用字串 `test`。

```
$test$ my statement $test$
```

如需 $ 符號引用的詳細資訊，請參閱 PostgreSQL 文件中的辭典結構下的「$ 符號引用的字串常數」。

python_program

傳回值的有效可執行 Python 程式。隨函數傳入的陳述式必須符合縮排要求，如 Python 網站上的《Python 程式碼格式指南》https://www.python.org/dev/peps/pep-0008/#indentation所述。如需詳細資訊，請參閱 UDF 的 Python 語言支援。

SQL_clause

SQL SELECT 子句。

SELECT 子句不可包含下列任何類型的子句：

- FROM
- INTO
- WHERE
- GROUP BY
- ORDER BY
- LIMIT

LANGUAGE { plpythonu | sql }

若是 Python，指定 `plpythonu`。若是 SQL，指定 `sql`。您必須具有 SQL 或 plpythonu 的語言使用權許可。如需詳細資訊，請參閱 UDF 安全與權限。

使用須知

巢狀函數

您可以從 SQL UDF 內呼叫另一個 SQL 使用者定義的函數 (UDF)。當您執行 CREATE FUNCTION 命令時，巢狀函數必須存在。Amazon Redshift 不會追蹤 UDF 的依賴關係，因此，如果您捨棄巢狀函數，Amazon Redshift 不會傳回錯誤。不過，若巢狀函數不存在，UDF 將會失敗。例如，以下函數會在 SELECT 子句中呼叫 `f_sql_greater` 函數。

```
create function f_sql_commission (float, float )
  returns float
stable
as $$
  select f_sql_greater ($1, $2)
$$ language sql;
```

UDF 安全與權限

若要建立 UDF，您必須具有 SQL 或 plpythonu (Python) 的語言使用權許可。根據預設，USAGE ON LANGUAGE SQL 會授予 PUBLIC。不過，您必須將 USAGE ON LANGUAGE PLPYTHONU 明確授予特定使用者或群組。

若要撤銷 SQL 的使用權，請先從 PUBLIC 撤銷使用權。然後僅將 SQL 使用權授予獲得許可建立 SQL UDF 的特定使用者或群組。下列範例會撤銷 PUBLIC 的 SQL 使用權，然後將使用權授予使用者群組 `udf_devs`。

```
revoke usage on language sql from PUBLIC;
grant usage on language sql to group udf_devs;
```

若要執行 UDF，您必須具有每個函數的執行許可。根據預設，新 UDF 的執行許可會授予 PUBLIC。若要限制使用權，請從 PUBLIC 撤銷函數的執行許可。然後將許可授予特定個人或群組。

下列範例會撤銷 PUBLIC 的函數 `f_py_greater` 執行許可，然後將使用權授予使用者群組 `udf_devs`。

```
revoke execute on function f_py_greater(a float, b float) from PUBLIC;
grant execute on function f_py_greater(a float, b float) to group udf_devs;
```

根據預設，超級使用者具備所有權限。

如需詳細資訊，請參閱 GRANT 及 REVOKE。

範例

純量 Python UDF 範例

下列範例會建立比較兩個整數並傳回較大值的 Python UDF。

```
create function f_py_greater (a float, b float)
  returns float
stable
as $$
  if a > b:
    return a
  return b
$$ language plpythonu;
```

下列範例會查詢 SALES 資料表，並呼叫新的 f_py_greater 函數來查詢 COMMISSION 或 20% 的 PRICEPAID，以較大者為準。

```
select f_py_greater (commission, pricepaid*0.20) from sales;
```

純量 SQL UDF 範例

下列範例會建立一個函數，比較兩個數字並傳回較大的值。

```
create function f_sql_greater (float, float)
  returns float
stable
as $$
  select case when $1 > $2 then $1
    else $2
  end
$$ language sql;
```

下列查詢會呼叫新的 f_sql_greater 函數來查詢 SALES 資料表，並傳回 COMMISSION 或 20% 的 PRICEPAID，以較大者為準。

```
select f_sql_greater (commission, pricepaid*0.20) from sales;
```

CREATE GROUP

定義新的使用者群組。只有超級使用者才能建立群組。

語法

```
CREATE GROUP group_name
[ [ WITH ] [ USER username ] [, ...] ]
```

參數

group_name

新使用者群組的名稱。開頭為兩個底線的群組名稱保留供 Amazon Redshift 內部使用。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

WITH

選用的語法，指出 CREATE GROUP 的其他參數。

USER

在群組中新增一個或多個使用者。

username

要新增至群組的使用者名稱。

範例

以下範例會建立名為 ADMIN_GROUP 的使用者群組，當中包含兩個使用者 ADMIN1 和 ADMIN2。

```
create group admin_group with user admin1, admin2;
```

CREATE IDENTITY PROVIDER

定義新的身分提供者。只有超級使用者可以建立身分提供者。

語法

```
CREATE IDENTITY PROVIDER identity_provider_name TYPE type_name
NAMESPACE namespace_name
[PARAMETERS parameter_string]
[APPLICATION_ARN arn]
[IAM_ROLE iam_role]
```

參數

identity_provider_name

新身分提供者的名稱。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

type_name

要與之連接的身分提供者。Azure 是目前唯一支援的身分提供者。

namespace_name

命名空間 這是身分提供者目錄的唯一速記識別碼。

parameter_string

包含正確格式化 JSON 物件的字串，其中包含身分提供者所需的參數和值。

arn

IAM 身分中心受管應用程式的 Amazon 資源名稱 (ARN)。此參數僅適用於身分識別提供者類型為時。 AWSIDC

阿姆角色

提供連線至 IAM 身分中心的許可的 IAM 角色。此參數僅適用於身分識別提供者類型為時。 AWSIDC

範例

下列範例會建立名為 oauth_standard 的身分提供者 (類型為 azure)，目的是與 Microsoft Azure Active Directory (AD) 建立通訊。

```
CREATE IDENTITY PROVIDER oauth_standard TYPE azure
NAMESPACE 'aad'
PARAMETERS '{"issuer":"https://sts.windows.net/2sdfdsf-d475-420d-b5ac-667adad7c702/",
"client_id":"87f4aa26-78b7-410e-bf29-57b39929ef9a",
"client_secret":"BUAH~ewrqewrqwerUUY^%tHe1oNZShoiU7",
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"]
}'
```

您可以將 IAM 身分中心受管應用程式與現有的佈建叢集或 Amazon Redshift 無伺服器工作群組連接。這讓您能夠透過 IAM 身分中心管理 Redshift 資料庫的存取權。若要這麼做，請執行類似下列範例的 SQL 命令。您必須是資料庫管理員。

```
CREATE IDENTITY PROVIDER "redshift-idc-app" TYPE AWSIDC
NAMESPACE 'awsidc'
APPLICATION_ARN 'arn:aws:sso::123456789012:application/ssoins-12345f67fe123d4/apl-
a0b0a12dc123b1a4'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyRedshiftRole';
```

在此情況下，應用程式 ARN 會識別要連線的受管理應用程式。您可以通過運行找到它SELECT * FROM SVV_IDENTITY_PROVIDERS;。

如需使用 CREATE IDENTITY PROVIDER 的相關資訊 (包括其他範例)，請參閱 Amazon Redshift 的原生身分提供者 (IdP) 聯合。如需有關從 Redshift 設定 IAM 身分中心連線的詳細資訊，請參閱 Connect Redshift 與 IAM 身分中心連線，為使用者提供單一登入體驗。

CREATE LIBRARY

安裝 Python 程式庫，使用者可在使用 CREATE FUNCTION 命令建立使用者定義的函數 (UDF) 時採用。使用者安裝的程式庫的總和大小不得超過 100 MB。

CREATE LIBRARY 無法在交易區塊內 (BEGIN … END) 執行。如需交易的相關資訊，請參閱 可序列化隔離。

Amazon Redshift 支援 Python 2.7 版。如需詳細資訊，請參閱 www.python.org。

如需詳細資訊，請參閱 匯入自訂 Python 程式庫模組。

所需權限

以下是 CREATE LIBRARY 所需的權限：

- 超級使用者
- 具有 CREATE LIBRARY 權限或具有指定語言權限的使用者

語法

```
CREATE [ OR REPLACE ] LIBRARY library_name LANGUAGE plpythonu
FROM
{ 'https://file_url'
| 's3://bucketname/file_name'
authorization
  [ REGION [AS] 'aws_region']
```

```
  IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
}
```

參數

OR REPLACE

指定已有相同名稱的程式庫存在時，取代現有程式庫。REPLACE 會立即遞交。如果倚賴程式庫的 UDF 同時執行，UDF 可能會失敗或傳回意外的結果，即使 UDF 是在交易內執行也一樣。您必須是擁有者或超級使用者才能取代程式庫。

library_name

要安裝的程式庫名稱。您建立的程式庫不可包含與 Python 標準程式庫模組或 Amazon Redshift 預先安裝 Python 模組同名的模組。如果現有的使用者安裝程式庫使用與所安裝程式庫相同的 Python 套件，則您必須先捨棄現有的程式庫，才能安裝新的程式庫。如需詳細資訊，請參閱 UDF 的 Python 語言支援。

LANGUAGE plpythonu

要使用的語言。Python (plpythonu) 是唯一支援的語言。Amazon Redshift 支援 Python 2.7 版。如需詳細資訊，請參閱 www.python.org。

FROM

程式庫檔案的位置。您可以指定 Amazon S3 儲存貯體和物件名稱，也可以指定 URL，以從公開網站下載檔案。程式庫必須封裝為 .zip 檔案。如需詳細資訊，請參閱 Python 文件中的建構和安裝 Python 模組。

https://file_url

從公開網站下載檔案的 URL。URL 最多可包含三個重新導向。以下是檔案 URL 的範例。

```
'https://www.example.com/pylib.zip'
```

s3://bucket_name/file_name

單一 Amazon S3 物件的路徑，物件當中包含程式庫檔案。以下是 Amazon S3 物件路徑的範例。

```
's3://mybucket/my-pylib.zip'
```

如果您指定 Amazon S3 儲存貯體，則必須同時提供有下載檔案許可之 AWS 使用者的登入資料。

> **Important**
>
> 如果 Amazon S3 儲存貯體與 Amazon Redshift 叢集不在同一個 AWS 區域，您必須使用區域選項來指定資料所在的 AWS 區域。aws_region 的值必須與 COPY 命令的REGION參數說明中表格中列出的 AWS 區域相符。

authorization

此子句指出叢集在身分驗證和授權存取包含程式庫檔案的 Amazon S3 儲存貯體時使用的方法。叢集必須有使用 LIST 和 GET 動作存取 Amazon S3 的許可。

授權的語法與 COPY 命令授權相同。如需詳細資訊，請參閱 授權參數。

```
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>'
```

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREATE LIBRARY 命令時與叢集關聯的 IAM 角色。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。如果指定 IAM_ROLE，則不能使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY、SESSION_TOKEN 或 CREDENTIALS。

或者，如果 Amazon S3 儲存貯體使用的是伺服器端加密，請提供 credentials-args 字串中的加密金鑰。如果您使用的是臨時安全登入資料，請提供 credentials-args 字串中的暫時字符。

如需詳細資訊，請參閱 暫時安全憑證。

REGION [AS] aws_region

Amazon S3 儲存貯體所在的 AWS 區域。當 Amazon S3 儲存貯體與 Amazon Redshift 叢集不在同一個 AWS 區域時，就需要使用區域。aws_region 的值必須與 COPY 命令的REGION參數說明中表格中列出的 AWS 區域相符。

根據預設，建立程式庫會假設 Amazon S3 儲存貯體與 Amazon Redshift 叢集位於相同的 AWS 區域。

範例

以下兩個範例會安裝 urlparse Python 模組，它會封裝成名為 `urlparse3-1.0.3.zip` 的檔案。

以下命令會從上傳到位於美國東部區域的 Amazon S3 儲存貯體的封裝，安裝名為 f_urlparse 的 UDF 程式庫。

```
create library f_urlparse
language plpythonu
from 's3://mybucket/urlparse3-1.0.3.zip'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
region as 'us-east-1';
```

下列範例會從網站上的程式庫檔案安裝名為 f_urlparse 的程式庫。

```
create library f_urlparse
language plpythonu
from 'https://example.com/packages/urlparse3-1.0.3.zip';
```

CREATE MASKING POLICY

建立新的動態資料遮罩政策，以模糊指定格式的資料。如需動態資料遮罩的相關資訊，請參閱 動態資料遮罩。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以建立遮罩政策。

語法

```
CREATE MASKING POLICY
   policy_name [IF NOT EXISTS]
   WITH (input_columns)
   USING (masking_expression);
```

參數

policy_name

遮罩政策的名稱。遮罩政策的名稱不能與資料庫中已存在的另一個遮罩政策相同。

input_columns

使用格式的資料欄名稱元組 (col1 類型、col2 類型 ...)。

資料欄名稱會用來做為遮罩表示式的輸入。資料欄名稱不一定要與要遮罩的資料欄名稱相符，但輸入和輸出資料類型必須相符。

masking_expression

用來轉換目標資料欄的 SQL 運算式。其可以使用資料操作函數 (例如字符串操作函數) 編寫，或與使用 SQL、Python 或 AWS Lambda編寫的使用者定義函數一起編寫。您可以針對具有多個輸出的遮罩政策包含資料欄運算式的元組。如果您使用常數做為遮罩運算式，則必須將其明確轉換為符合輸入類型的類型。

對於在遮罩運算式中使用的任何使用者定義函數，您都必須擁有 USAGE 權限。

CREATE MATERIALIZED VIEW

根據一個或多個 Amazon Redshift 資料表建立具體化視觀表。您也可以讓具體化視觀表以使用 Spectrum 或聯合查詢建立的外部資料表為依據。如需 Spectrum 的詳細資訊，請參閱使用 Amazon Redshift Spectrum 查詢外部資料。如需聯合查詢的詳細資訊，請參閱 使用 Amazon Redshift 中的聯合查詢來查詢資料。

語法

```
CREATE MATERIALIZED VIEW mv_name
[ BACKUP { YES | NO } ]
[ table_attributes ]
[ AUTO REFRESH { YES | NO } ]
AS query
```

參數

BACKUP

此子句指定是否將具體化視觀表包含在自動化和手動叢集快照中，這些快照會儲存在 Amazon S3 內。

BACKUP 的預設值為 YES。

您可以指定 BACKUP NO 來減少建立快照及從快照還原的處理時間，以及減少 Amazon S3 中所需要的儲存體空間。

> **Note**
>
> BACKUP NO 設定對於將資料自動複寫到叢集內其他節點的操作並無影響，因此指定了 BACKUP NO 的資料表會在節點故障時還原。

table_attributes

此子句指定如何分配具體化檢視中的資料，其中包括：

- 具體化檢視的分佈樣式，其格式為 DISTSTYLE { EVEN | ALL | KEY }。如果您省略此子句，則分佈樣式為 EVEN。如需詳細資訊，請參閱 分佈樣式。
- 具體化檢視的分佈索引鍵，其格式為 DISTKEY (*distkey_identifier*)。如需詳細資訊，請參閱 指定分佈樣式。
- 具體化檢視的排序索引鍵，其格式為 SORTKEY (*column_name* [, ...])。如需詳細資訊，請參閱 使用排序索引鍵。

AS query

定義具體化視觀表及其內容的有效 SELECT 陳述式。查詢的結果集會定義具體化檢視的資料行與資料列。如需有關建立具體化檢視時有何限制的資訊，請參閱限制。

此外，查詢中使用的特定 SQL 語言建構會決定具體化檢視是否可以增量或完全重新整理。如需重新整理方法的相關資訊，請參閱REFRESH MATERIALIZED VIEW。如需增量重新整理之限制的相關資訊，請參閱增量重新整理的限制。

如果查詢包含不支援增量重新整理的 SQL 命令，則 Amazon Redshift 會顯示訊息，指出具體化視觀表將使用完整重新整理。不一定會顯示訊息，視 SQL 用戶端應用程式而定。檢查 STV_MV_INFO 的 state 資料欄，以查看具體化檢視所使用的重新整理類型。

AUTO REFRESH

該子句會定義是否應使用基本資料表中的最新變更自動重新整理具體化視觀表。預設值為 NO。如需詳細資訊，請參閱 重新整理具體化視觀表。

使用須知

若要建立具體化檢視，您必須具有下列權限：

- 結構描述的 CREATE 權限。
- 在基礎資料表上建立具體化視觀表的資料表層級或資料欄層級 SELECT 權限。如果您擁有特定資料欄層級的權限，就可以僅針對這些資料欄建立具體化視觀表。

資料命名中具體化視觀表的累加式重新整理

Amazon Redshift 支援在共用基礎資料表時，針對消費者資料保護中的具體化視圖進行自動和累加式重新整理。累加式重新整理是一項操作，Amazon Redshift 會識別基礎資料表或上次重新整理後發生的資

料表中的變更，並僅更新具體化視觀表中的對應記錄。這樣的執行速度比完全重新整理更快，並提高工作負載效能。您不必變更具體化檢視的定義，即可利用累加式重新整理。

利用具體化視觀表的累加式重新整理時，有幾個限制需要注意：

- 具體化視觀表必須只參照一個資料庫，無論是本機或遠端資料庫。
- 累加式重新整理只能在新具體化視觀表上使用。因此，您必須刪除現有的具體化視觀表並重新建立它們，以進行累加式重新整理

如需有關在資料清單中建立具體化視觀表的詳細資訊，請參閱在 Amazon Redshift 資料共用中使用檢視，其中包含數個查詢範例。

具體化檢視或基礎資料表的 DDL 更新

在 Amazon Redshift 中使用具體化視觀表時，請遵循下列對具體化檢視或基礎資料表進行資料定義語言 (DDL) 更新時的使用注意事項。

- 您可以將資料欄新增到基礎資料表，而不影響任何參考該基礎資料表的具體化視觀表。
- 某些操作也可能使具體化檢視進入完全無法重新整理的狀態。例如：重新命名或卸除資料欄、變更資料欄類型，以及變更結構描述名稱等操作。這類具體化檢視可以進行查詢，但無法重新整理。在此情況下，您必須卸除並重新建立具體化檢視。
- 一般而言，您無法修改具體化檢視的定義 (其 SQL 陳述式)。
- 您無法將具體化檢視重新命名。

限制

您無法定義參考或包含以下任何一項的具體化檢視：

- 標準檢視，或系統資料表和檢視。
- 暫存資料表。
- 使用者定義的函數。
- ORDER BY、LIMIT 或 OFFSET 子句。
- 對基礎資料表的近期繫結參考。換句話說，定義具體化檢視的 SQL 查詢中所參考的任何基底資料表或相關資料行都必須存在，且必須有效。
- 僅限領導節點的函數：CURRENT_SCHEMA、CURRENT_SCHEMAS、HAS_DATABASE_PRIVILEGE、HAS_SCHEMA_PRIVIL

- 具體化視觀表定義包含可變函數或外部結構描述時，您無法使用 AUTO REFRESH YES 選項。當您在另一個具體化視觀表上定義具體化視觀表時，也無法使用此選項。
- 您不需要在具體化視觀表上手動執行 ANALYZE。目前這種情況只會透過 AUTO ANALYZE 發生。如需詳細資訊，請參閱 分析資料表。

範例

下列範例會從三個加入和彙總的基礎資料表建立具體化視觀表。每一列代表具有售票數的類別。查詢 tickets_mv 具體化檢視時，您可以直接存取 tickets_mv 具體化檢視中的預先計算資料。

```
CREATE MATERIALIZED VIEW tickets_mv AS
    select   catgroup,
    sum(qtysold) as sold
    from     category c, event e, sales s
    where    c.catid = e.catid
    and      e.eventid = s.eventid
    group by catgroup;
```

下列範例會建立類似上述範例的具體化視觀表，並使用彙總函數 MAX()。

```
CREATE MATERIALIZED VIEW tickets_mv_max AS
    select   catgroup,
    max(qtysold) as sold
    from     category c, event e, sales s
    where    c.catid = e.catid
    and      e.eventid = s.eventid
    group by catgroup;

SELECT name, state FROM STV_MV_INFO;
```

下列範例使用 UNION ALL 子句加入 Amazon Redshift `public_sales` 資料表與 Redshift Spectrum `spectrum.sales` 資料表來建立具體化視觀表 `mv_sales_vw`。如需 Amazon Redshift Spectrum 的 CREATE EXTERNAL TABLE 命令的詳細資訊，請參閱 CREATE EXTERNAL TABLE。Redshift Spectrum 外部資料表會參考 Amazon S3 上的資料。

```
CREATE MATERIALIZED VIEW mv_sales_vw as
select salesid, qtysold, pricepaid, commission, saletime from public.sales
union all
select salesid, qtysold, pricepaid, commission, saletime from spectrum.sales
```

下列範例會根據聯合查詢外部資料表建立具體化檢視 mv_fq。如需聯合查詢的詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

```
CREATE MATERIALIZED VIEW mv_fq as select firstname, lastname from apg.mv_fq_example;

select firstname, lastname from mv_fq;
 firstname | lastname
-----------+----------
 John      | Day
 Jane      | Doe
(2 rows)
```

下列範例顯示具體化檢視的定義。

```
SELECT pg_catalog.pg_get_viewdef('mv_sales_vw'::regclass::oid, true);

pg_get_viewdef
-------------------------------------------------
create materialized view mv_sales_vw as select a from t;
```

下列範例顯示如何在具體化視觀表定義中設定 AUTO REFRESH，以及如何指定 DISTSTYLE。首先，建立一個簡單的基底資料表。

```
CREATE TABLE baseball_table (ball int, bat int);
```

然後建立具體化視觀表。

```
CREATE MATERIALIZED VIEW mv_baseball DISTSTYLE ALL AUTO REFRESH YES AS SELECT ball AS
 baseball FROM baseball_table;
```

現在您可以查詢 mv_baseball 具體化視觀表。若要檢查具體化視觀表是否已開啟 AUTO REFRESH，請參閱 STV_MV_INFO。

下列範例會建立參照另一個資料庫中來源資料表的具體化視觀表。假設包含來源資料表的資料庫 database_A 與您在 database_B 中建立的具體化視觀表位於相同叢集或工作群組中。(您可以將範例取代為自己的資料庫。) 首先，在 database_A 中建立一個名為 cities 的資料表，其中包含 cityname 資料欄。使資料欄的資料類型成為 VARCHAR。建立來源資料表之後，在 database_B 中執行下列命令，以建立其來源為 cities 資料表的具體化視觀表。確實在 FROM 子句中指定來源資料表的資料庫和結構描述：

```
CREATE MATERIALIZED VIEW cities_mv AS
SELECT  cityname
FROM    database_A.public.cities;
```

查詢您建立的具體化視觀表。查詢會擷取其原始來源為 database_A 中 cities 資料表的記錄：

```
select * from cities_mv;
```

當您執行 SELECT 陳述式時，cities_mv 會傳回記錄。只有在執行 REFRESH 陳述式時，才會從來源資料表重新整理記錄。此外，請注意，您無法直接在具體化視觀表中更新記錄。如需有關重新整理具體化視觀表中資料的資訊，請參閱 REFRESH MATERIALIZED VIEW。

如需有關具體化檢視概觀，以及用來重新整理和捨棄具體化檢視之 SQL 命令的詳細資訊，請參閱下列主題：

- 在 Amazon Redshift 中建立具體化視觀表
- REFRESH MATERIALIZED VIEW
- DROP MATERIALIZED VIEW

CREATE MODEL

主題

必要條件

使用 CREATE MODEL 陳述式之前，請先完成 使用 Amazon Redshift ML 的叢集設定 中的先決條件。以下是先決條件的概要。

- 使用 AWS 管理主控台或 AWS 命令列界面 (AWS CLI) 建立 Amazon Redshift 叢集。
- 建立叢集時附加 AWS Identity and Access Management (IAM) 政策。
- 若要允許 Amazon Redshift 並 SageMaker 假設該角色與其他服務互動，請將適當的信任政策新增至 IAM 角色。

如需 IAM 角色、信任政策和其他先決條件的詳細資訊，請參閱 使用 Amazon Redshift ML 的叢集設定。

接下來，你可以找到 CREATE MODEL 陳述式的不同使用案例。

- 簡易 CREATE MODEL
- CREATE MODEL 和使用者指引
- CREATE XGBoost 模型與 AUTO OFF
- 使用自有模型 (BYOM) - 本機推論
- CREATE MODEL 與 K-MEANS
- 完整的 CREATE MODEL

所需權限

以下是 CREATE MODEL 所需的權限：

- 超級使用者
- 具有 CREATE MODEL 權限的使用者
- 具有 GRANT CREATE MODEL 權限的角色

成本控制

Amazon Redshift ML 會使用現有的叢集資源建立預測模型，因此您不必支付額外費用。但是，如果您需要調整叢集大小或想要訓練模型，則可能會產生額外費用。Amazon Redshift ML 使用 Amazon SageMaker 來訓練模型，這確實會產生額外的相關成本。有一些方法可以控制額外成本，例如限制訓練所耗用的時間上限，或限制用於訓練模型的訓練範例數量。如需詳細資訊，請參閱使用 Amazon Redshift 資料 API。

完整的 CREATE MODEL

以下概述完整 CREATE MODEL 語法的基本選項。

完整的 CREATE MODEL 語法

以下是 CREATE MODEL 陳述式的完整語法。

> **Important**
>
> 使用 CREATE MODEL 陳述式建立模型時，請遵循下列語法中的關鍵字順序。

```
CREATE MODEL model_name
    FROM { table_name | ( select_statement )  | 'job_name' }
    [ TARGET column_name ]
    FUNCTION function_name ( data_type [, ...] )
    [ RETURNS super ]
    IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
    [ AUTO ON / OFF ]
      -- default is AUTO ON
    [ MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER | KMEANS | FORECAST } ]
      -- not required for non AUTO OFF case, default is the list of all supported types
      -- required for AUTO OFF
    [ PROBLEM_TYPE ( REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION ) ]
      -- not supported when AUTO OFF
    [ OBJECTIVE ( 'MSE' | 'Accuracy' | 'F1' | 'F1_Macro' | 'AUC' |
                  'reg:squarederror' | 'reg:squaredlogerror'| 'reg:logistic'|
                  'reg:pseudohubererror' | 'reg:tweedie' | 'binary:logistic' |
 'binary:hinge',
                  'multi:softmax' | 'RMSE' | 'WAPE' | 'MAPE' | 'MASE' |
 'AverageWeightedQuantileLoss' ) ]
       -- for AUTO ON: first 5 are valid
       -- for AUTO OFF: 6-13 are valid
       -- for FORECAST: 14-18 are valid
    [ PREPROCESSORS 'string' ]
      -- required for AUTO OFF, when it has to be 'none'
      -- optional for AUTO ON
    [ HYPERPARAMETERS { DEFAULT | DEFAULT EXCEPT ( Key 'value' (,...) ) } ]
      -- support XGBoost hyperparameters, except OBJECTIVE
      -- required and only allowed for AUTO OFF
      -- default NUM_ROUND is 100
      -- NUM_CLASS is required if objective is multi:softmax (only possible for AUTO
 OFF)
     [ SETTINGS (
       S3_BUCKET 'bucket',  |
        -- required
```

```
    TAGS 'string', |
      -- optional
    KMS_KEY_ID 'kms_string', |
      -- optional
    S3_GARBAGE_COLLECT on / off, |
      -- optional, defualt is on.
    MAX_CELLS integer, |
      -- optional, default is 1,000,000
    MAX_RUNTIME integer (, ...) |
      -- optional, default is 5400 (1.5 hours)
    HORIZON integer, |
      -- required if creating a forecast model
    FREQUENCY integer, |
      -- required if creating a forecast model
    PERCENTILES string
      -- optional if creating a forecast model
  ) ]
```

參數

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

FROM { table_name | (select_query) | 'job_name'}

指定訓練資料的 table_name 或查詢。這可以是系統中現有的資料表，也可以是與 Amazon RedShift 相容的 SELECT 查詢，並以括號括住，也就是 ()。查詢結果中至少必須有兩個資料列。

TARGET column_name

成為預測目標的資料欄名稱。該資料欄必須存在於 FROM 子句中。

FUNCTION function_name (data_type [, ...])

要建立的函數名稱和輸入引數的資料類型。您可以在資料庫中提供結構描述的結構描述名稱，而不是函數名稱。

RETURNS SUPER (預覽)

從模型傳回的資料類型。傳回的 SUPER 資料類型僅適用於遠端 BYOM 模型。

IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREAT MODEL 命令時與叢集關聯的 IAM 角色。或者，您可以指定 IAM 角色的 ARN 以使用該角色。

[AUTO ON / OFF]

開啟或關閉預處理器、演算法和超參數選擇的 CREATE MODEL 自動探索。在建立 Forecast 模型時指定開啟表示使用 AutoPredictor，其中 Amazon Forecast 會將最佳演算法組合套用至資料集中的每個時間序列。

MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER | KMEANS | FORECAST }

(選擇性) 指定模型類型。您可以指定是否要訓練特定模型類型的模型，例如 XGBoost、多層感知器 (MLP)、KMEANS 或線性學習器，這些都是 Amazon 自動駕駛儀支援的演算法。 SageMaker 如果未指定參數，則會在訓練期間搜尋所有支援的模型類型，以取得最佳模型。您也可以在 Redshift ML 中建立預測模型，以建立準確的時間序列預測。

PROBLEM_TYPE (REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION)

(選擇性) 指定問題類型。如果您知道問題類型，則可以將 Amazon Redshift 限制為僅搜尋該特定模型類型的最佳模型。如果未指定此參數，則訓練期間會根據您的資料探索問題類型。

目的 ('MSE' | '準確性' | 'F1' | 'F1 宏' | 'REC' | 'REG：方形錯誤' | '註冊：物流' | '註冊：偽大師' | 'REG：' 註釋：鑷子 '|「二進制：後勤」|「二進制：鉸鏈」|「RMSE」|「二進制：鉸鏈」|「RMSE」APE' | 'MAPE' | '馬斯' | ') AverageWeightedQuantileLoss

(選擇性) 指定用來測量機器學習系統預測品質的目標指標名稱。此指標會在訓練期間進行最佳化，從資料中提供模型參數值的最佳估計值。如果您沒有明確指定指標，則預設行為是自動將 MSE: 用於迴歸、將 F1: 用於二進制分類、將 Accuracy: 用於多類別分類。如需有關目標的詳細資訊，請參閱 Amazon SageMaker API 參考和 XGBOOST 文件中的學習任務參數中的 AutoML JobObjective。值 RMSE，WAPE，MAPE，MASE，並且 AverageWeightedQuantileLoss 僅適用於 Forecast 模型。如需詳細資訊，請參閱CreateAuto預測值 API 作業。

PREPROCESSORS 'string'

(選擇性) 指定特定資料欄集的特定預處理器組合。格式會是 columnSets 清單，以及要套用到每一組資料欄的適當轉換。Amazon Redshift 將特定變壓器列表中的所有變壓器應用於相應 ColumnSet 的所有列。例如，要使用輸入器應 OneHotEncoder 用於列 t1 和 t2，請使用以下示例命令。

```
CREATE MODEL customer_churn
FROM customer_data
TARGET 'Churn'
FUNCTION predict_churn
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
PROBLEM_TYPE BINARY_CLASSIFICATION
```

```
OBJECTIVE 'F1'
PREPROCESSORS '[
...
  {"ColumnSet": [
      "t1",
      "t2"
    ],
    "Transformers": [
      "OneHotEncoder",
      "Imputer"
    ]
  },
  {"ColumnSet": [
      "t3"
    ],
    "Transformers": [
      "OneHotEncoder"
    ]
  },
  {"ColumnSet": [
      "temp"
    ],
    "Transformers": [
      "Imputer",
      "NumericPassthrough"
    ]
  }
]'
SETTINGS (
  S3_BUCKET 'bucket'
)
```

HYPERPARAMETERS { DEFAULT | DEFAULT EXCEPT (key ‘value’ (,..)) }

指定使用預設的 XGBoost 參數或以使用者指定的值覆寫。這些值必須用單引號括住。以下是 XGBoost 及其預設值的參數範例。

參數名稱	參數值	預設值	備註
num_class	Integer	多類別分	N/A

參數名稱	參數值	預設值	備註
		類所需。	
num_round	Integer	100	N/A
tree_method	字串	Auto	N/A
max_depth	Integer	6	[0 , 10]
min_child_weight	Float	1	MinValue: 0, MaxValue
子範例	Float	1	MinValue MaxValue：半點
Gamma	Float	0	MinValue: 0, MaxValue: 5
alpha	Float	0	MinValue: 0, MaxValue
eta	Float	0.3	MinValue：零 MaxValue
colsample_byleve	Float	1	MinValue：零一 MaxValue
colsample_bynode	Float	1	MinValue：零一 MaxValue
colsample_bytree	Float	1	MinValue MaxValue：半點
lambda	Float	1	MinValue: 0, MaxValue
max_delta_step	Integer	0	[0, 10]

SETTINGS (S3_BUCKET 'bucket', | TAGS 'string', | KMS_KEY_ID 'kms_string' , | S3_GARBAGE_COLLECT on / off, | MAX_CELLS integer , | MAX_RUNTIME (,...) , | HORIZON integer, | FREQUENCY forecast_frequency, | PERCENTILES array of strings)

S3_BUCKET 子句會指定用來儲存中繼結果的 Amazon S3 位置。

(選用) TAGS 參數是以逗號分隔的索引鍵值配對清單，可用來標記在 Amazon 中建立的資源 SageMaker；以及 Amazon Forecast。標籤可協助您整理資源和配置成本。配對中的值是可選的，因此您可以使用 `key=value` 格式或直接建立索引鍵來建立標籤。如需 Amazon Redshift 中標籤的相關資訊，請參閱標記概觀。

(選擇性) KMS_KEY_ID 會指定 Amazon Redshift 是否使用伺服器端加密搭配 AWS KMS 索引鍵來保護靜態資料。傳輸中的資料會受到 Secure Sockets Layer (SSL) 保護。

(選擇性) S3_GARBAGE_COLLECT { ON | OFF } 會指定 Amazon Redshift 是否對用於訓練模型的結果資料集和模型執行垃圾回收。如果設定為 OFF，用於訓練模型的結果資料集和模型會保留在 Amazon S3 中，並可用於其他用途。如果設定為 ON，Amazon Redshift 會在訓練完成後刪除 Amazon S3 中的成品。預設值為 ON。

(選擇性) MAX_CELLS 會指定訓練資料中的儲存格數目。此值是記錄數目 (訓練查詢或資料表中) 乘以欄數的乘積。預設值為 1,000,000。

(選擇性) MAX_RUNTIME 會指定訓練的時間上限。視資料集大小而定，訓練工作通常會更快完成。這會指定訓練應耗用的時間上限。預設值為 5,400 (90 分鐘)。

HORIZON 會指定預測模型可傳回的預測數量上限。模型一經過訓練，就無法再變更此整數。如果訓練預測模型，則需要此參數。

FREQUENCY 會指定您希望預測使用多細微的時間單位。可用選項為 `Y | M | W | D | H | 30min | 15min | 10min | 5min | 1min`。如果訓練預測模型，則需要此參數。

(選擇性) PERCENTILES 是逗號分隔的字串，可指定用來訓練預測器的預測類型。預測類型可以是 0.01 到 0.99 之間的分位數，增量單位為 0.01 或更高。您也可以使用平均值指定平均預測。您最多可以指定五種預測類型。

使用須知

使用 CREATE MODEL 時，請考量下列事項。

- CREATE MODEL 陳述式會以非同步模式運作，並在將訓練資料匯出至 Amazon S3 時傳回。Amazon 培訓的其餘步驟 SageMaker 在後台進行。訓練正在進行中時，對應的推論函數會顯示，但無法執行。您可以查詢 STV_ML_MODEL_INFO 以查看訓練狀態。
- 根據預設，在自動模式中，訓練最多可在背景執行 90 分鐘，而且可以延長。若要取消訓練，只要執行 DROP MODEL 命令即可。
- 您用來建立模型的 Amazon Redshift 叢集，以及用來暫存訓練資料和模型成品的 Amazon S3 儲存貯體必須位於相同 AWS 區域。
- 在模型訓練期間，Amazon Redshift 會將中繼成品 SageMaker 存放在您提供的 Amazon S3 儲存貯體中。根據預設，Amazon Redshift 會在 CREATE MODEL 操作結束時執行垃圾回收。Amazon Redshift 會從 Amazon S3 中移除這些物件。若要將這些成品保留在 Amazon S3 上，請設定 S3_GARBAGE COLLECT OFF 選項。
- 您必須在 FROM 子句中提供的訓練資料中使用至少 500 個資料列。
- 使用 CREATE MODEL 陳述式時，您最多只能在 FROM { table_name | (select_query) } 子句中指定 256 個特徵 (輸入) 資料欄。
- 若是 AUTO ON，您可以用來做為訓練集的資料欄類型為 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE、BOOLEAN、CHAR、VARCHAR、DAT 和 TIMESTAMPTZ。若是 AUTO OFF，您可以用來做為訓練集的資料欄類型為 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE 和 BOOLEAN。
- 您不能使用 DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP、TIMESTAMPTZ、GEOMETRY、GEOGRAPHY、HI 或 VARBYTE 做為目標資料欄類型。
- 若要改善模型精確度，請執行下列其中一項操作：
  - 當您在 FROM 子句中指定訓練資料時，在 CREATE MODEL 命令中盡可能多新增相關資料欄。
  - 使用較大的值做為 MAX_RUNTIME 和 MAX_CELLS。此參數的值越大，訓練模型的成本也會增加。
- 只要計算訓練資料並將其匯出至 Amazon S3 儲存貯體，就會立即傳回 CREATE MODEL 陳述式執行。在此之後，您可以使用 SHOW MODEL 命令檢查訓練的狀態。在背景訓練的模型失敗時，您可以使用 SHOW MODEL 來檢查錯誤。您無法重試失敗的模型。使用 DROP MODEL 可移除失敗的模型並重新建立新模型。如需 SHOW MODEL 的相關資訊，請參閱 SHOW MODEL。
- 本機 BYOM 支援 Amazon Redshift ML 在非 BYOM 案例中支援的同類型模型。Amazon Redshift 支援一般 XGBoost (使用 1.0 版或更新版本)、不含預處理器的 KMEANS 模型，以及由 Amazon 自動駕駛訓練的 XGBoost /MLP/ 線性學習模型。 SageMaker 它使用 Autopilot 指定的預處理器支持後者，該處理器也受到 Amazon SageMaker Neo 的支持。

- 如果您的 Amazon Redshift 叢集已啟用虛擬私有雲端 (VPC) 的增強型路由功能，請務必為叢集所在的虛擬私有雲端建立 Amazon S3 SageMaker VPC 端點和 VPC 端點。這樣做可在 CREATE MODEL 期間，讓流量在這些服務之間通過您的 VPC。如需詳細資訊，請參閱SageMaker 澄清 Job Amazon VPC 子網路和安全群組。

使用案例

以下使用案例會示範如何使用 CREATE MODEL 來滿足您的需求。

簡易 CREATE MODEL

以下概述 CREATE MODEL 語法的基本選項。

簡易 CREATE MODEL 語法

```
CREATE MODEL model_name
    FROM { table_name | ( select_query ) }
    TARGET column_name
    FUNCTION prediction_function_name
    IAM_ROLE { default }
    SETTINGS (
      S3_BUCKET 'bucket',
      [ MAX_CELLS integer ]
    )
```

簡易 CREATE MODEL 參數

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

FROM { table_name | (select_query) }

指定訓練資料的 table_name 或查詢。這可以是系統中現有的資料表，也可以是與 Amazon RedShift 相容的 SELECT 查詢，並以括號括住，也就是 ()。查詢結果中至少必須有兩個資料列。

TARGET column_name

成為預測目標的資料欄名稱。該資料欄必須存在於 FROM 子句中。

FUNCTION prediction_function_name

該值會指定要由 CREATE MODEL 產生的 Amazon Redshift 機器學習函數名稱，並用於使用此模型進行預測。該函數會在與模型物件相同的結構描述中建立，並且可以多載。

Amazon Redshift 機器學習可支援模型，例如用於迴歸和分類的 Xtreme 梯度提升樹 (XGBoost) 模型。

IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREAT MODEL 命令時與叢集關聯的 IAM 角色。或者，您也可以指定 IAM 角色的 ARN 以使用該角色。

S3_BUCKET 'bucket'

您先前建立的 Amazon S3 儲存貯體名稱，用於在 Amazon Redshift 和. 之間共用訓練資料和 SageMaker成品。Amazon Redshift 會在卸載訓練資料之前，在此儲存貯體中建立一個子資料夾。訓練完成後，Amazon Redshift 會刪除建立的子資料夾及其內容。

MAX_CELLS 整數

要從 FROM 子句匯出的儲存格數目上限。預設值為 1,000,000。

儲存格數目是訓練資料 (由 FROM 子句資料表或查詢產生) 中的資料列數乘以資料欄數的乘積。如果訓練資料中的儲存格數目大於 max_cell 參數所指定的儲存格數目，CREATE MODEL 會縮減 FROM 子句訓練資料，將訓練集的大小縮減到低於 MAX_CELLS。允許更大的訓練資料集可以產生更高的準確性，但也可能意味著模型需要更長的訓練時間和更高的成本。

如需 Amazon Redshift 使用成本的資訊，請參閱 使用 Amazon Redshift ML 的成本。

如需各種儲存格數目的成本和免費試用的相關資訊，請參閱 Amazon Redshift 定價。

CREATE MODEL 和使用者指引

以下說明 簡易 CREATE MODEL 中所述選項之外的 CREATE MODEL 選項。

根據預設，CREATE MODEL 會搜尋特定資料集的最佳預處理和模型組合。您可能需要模型的額外控制項目或對其引入其他領域知識 (例如問題類型或目標)。在客戶流失案例中，如果「客戶不活躍」的結果很少見，則 F1 目標通常高於準確度目標。因為高準確度模型可能會一直預測到「客戶很活躍」，此結果雖然具有高準確度，但商業價值很小。如需 F1 目標的相關資訊，請參閱 Amazon SageMaker API 參考JobObjective中的 AutoML。

然後，CREATE MODEL 會遵循您在指定層面的建議，例如目標。同時，CREATE MODEL 會自動發現最佳的預處理器和最佳的超參數。

CREATE MODEL 和使用者指引語法

CREATE MODEL 可為您可以指定的層面及 Amazon Redshift 可自動發現的層面提供更大的彈性。

```
CREATE MODEL model_name
    FROM { table_name | ( select_statement ) }
    TARGET column_name
    FUNCTION function_name
    IAM_ROLE { default }
    [ MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER} ]
    [ PROBLEM_TYPE ( REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION ) ]
    [ OBJECTIVE ( 'MSE' | 'Accuracy' | 'F1' | 'F1Macro' | 'AUC') ]
    SETTINGS (
      S3_BUCKET 'bucket', |
      S3_GARBAGE_COLLECT { ON | OFF }, |
      KMS_KEY_ID 'kms_key_id', |
      MAX_CELLS integer, |
      MAX_RUNTIME integer (, ...)
    )
```

CREATE MODEL 和使用者指引參數

MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER }

(選擇性) 指定模型類型。您可以指定是否要訓練特定模型類型的模型，例如 XGBoost、多層感知器 (MLP) 或線性學習器，這些都是 Amazon Autopilot 支援的演算法。 SageMaker 如果未指定參數，則會在訓練期間搜尋所有支援的模型類型，以取得最佳模型。

PROBLEM_TYPE (REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION)

(選擇性) 指定問題類型。如果您知道問題類型，則可以將 Amazon Redshift 限制為僅搜尋該特定模型類型的最佳模型。如果未指定此參數，則訓練期間會根據您的資料探索問題類型。

OBJECTIVE ('MSE' | 'Accuracy' | 'F1' | 'F1Macro' | 'AUC')

(選擇性) 指定用來測量機器學習系統預測品質的目標指標名稱。此指標會在訓練期間進行最佳化，從資料中提供模型參數值的最佳估計值。如果您沒有明確指定指標，則預設行為是自動將 MSE: 用於迴歸、將 F1: 用於二進制分類、將 Accuracy: 用於多類別分類。如需有關目標的詳細資訊，請參閱 Amazon SageMaker API 參考JobObjective中的 AutoML。

MAX_CELLS 整數

(選擇性) 指定訓練資料中的儲存格數目。此值是記錄數目 (訓練查詢或資料表中) 乘以欄數的乘積。預設值為 1,000,000。

MAX_RUNTIME 整數

(選擇性) 指定訓練的時間上限。視資料集大小而定，訓練工作通常會更快完成。這會指定訓練應耗用的時間上限。預設值為 5,400 (90 分鐘)。

S3_GARBAGE_COLLECT { ON | OFF }

(選擇性) 指定 Amazon Redshift 是否對用於訓練模型的結果資料集和模型執行垃圾回收。如果設定為 OFF，用於訓練模型的結果資料集和模型會保留在 Amazon S3 中，並可用於其他用途。如果設定為 ON，Amazon Redshift 會在訓練完成後刪除 Amazon S3 中的成品。預設值為 ON。

KMS_KEY_ID 'kms_key_id'

(選擇性) 指定 Amazon Redshift 是否使用伺服器端加密搭配 AWS KMS 索引鍵來保護靜態資料。傳輸中的資料會受到 Secure Sockets Layer (SSL) 保護。

PREPROCESSORS 'string'

(選擇性) 指定特定資料欄集的特定預處理器組合。格式會是 columnSets 清單，以及要套用到每一組資料欄的適當轉換。Amazon Redshift 將特定變壓器列表中的所有變壓器應用於相應 ColumnSet 的所有列。例如，要使用輸入器應 OneHotEncoder 用於列 t1 和 t2，請使用以下示例命令。

```
CREATE MODEL customer_churn
FROM customer_data
TARGET 'Churn'
FUNCTION predict_churn
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
PROBLEM_TYPE BINARY_CLASSIFICATION
OBJECTIVE 'F1'
PREPROCESSORS '[
...
{"ColumnSet": [
    "t1",
    "t2"
  ],
  "Transformers": [
    "OneHotEncoder",
    "Imputer"
  ]
},
{"ColumnSet": [
    "t3"
  ],
  "Transformers": [
    "OneHotEncoder"
```

```
  ]
},
{"ColumnSet": [
    "temp"
  ],
  "Transformers": [
    "Imputer",
    "NumericPassthrough"
  ]
}
]'
SETTINGS (
S3_BUCKET 'bucket'
)
```

Amazon Redshift 支援以下轉換：

- OneHotEncoder — 通常用於將離散值編碼為具有一個非零值的二進位向量。此轉換適用於許多機器學習模型。
- OrdinalEncoder — 將離散值編碼為單一整數。此轉換適用於特定機器學習模型，例如 MLP 和線性學習程式。
- NumericPassthrough — 將輸入原樣傳遞到模型中。
- Imputer – 填入遺漏值，而不是數字 (NaN) 值。
- ImputerWithIndicator — 填入缺少的值和 NaN 值。此轉換也會建立指出是否有任何值遺失和填入的指示器。
- Normalizer – 將值標準化，可改善許多機器學習演算法的效能。
- DateTimeVectorizer — 建立向量內嵌，代表可在機器學習模型中使用的日期時間資料類型欄。
- PCA – 將資料投影到較低的維度空間中，以減少特徵數量，同時盡可能保留更多資訊。
- StandardScaler — 透過移除平均值和縮放為單位差異來標準化特徵。
- MinMax — 透過將每個特徵縮放至指定範圍來轉換特徵。

Amazon Redshift ML 會儲存經過訓練的轉換，並在預測查詢中自動套用這些轉換。從模型產生預測時，您不需要指定這些轉換。

CREATE XGBoost 模型與 AUTO OFF

AUTO OFF CREATE MODEL 在預設的 CREATE MODEL 中通常有不同的目標。

如果進階使用者已知道想要的模型類型，以及訓練這些模型時所使用的超參數，您可以使用搭配 AUTO OFF 的 CREATE MODEL 來關閉預處理器和超參數的 CREATE MODEL 自動探索功能。若要這樣做，您必須明確指定模型類型。當 AUTO 設定為 OFF 時，XGBoost 是目前唯一支援的模型類型。您可以指定超參數。Amazon Redshift 會針對您指定的任何超參數使用預設值。

CREATE XGBoost 模型與 AUTO OFF 語法

```
CREATE MODEL model_name
    FROM { table_name | (select_statement ) }
    TARGET column_name
    FUNCTION function_name
    IAM_ROLE { default }
    AUTO OFF
    MODEL_TYPE XGBOOST
    OBJECTIVE { 'reg:squarederror' | 'reg:squaredlogerror' | 'reg:logistic' |
                'reg:pseudohubererror' | 'reg:tweedie' | 'binary:logistic' |
 'binary:hinge' |
                'multi:softmax' | 'rank:pairwise' | 'rank:ndcg' }
    HYPERPARAMETERS DEFAULT EXCEPT (
        NUM_ROUND '10',
        ETA '0.2',
        NUM_CLASS '10',
        (, ...)
    )
    PREPROCESSORS 'none'
    SETTINGS (
      S3_BUCKET 'bucket', |
      S3_GARBAGE_COLLECT { ON | OFF }, |
      KMS_KEY_ID 'kms_key_id', |
      MAX_CELLS integer, |
      MAX_RUNTIME integer (, ...)
    )
```

CREATE XGBoost 模型與 AUTO OFF 參數

AUTO OFF

關閉預處理器、演算法和超參數選擇的 CREATE MODEL 自動探索。

MODEL_TYPE XGBOOST

指定使用 XGBOOST 來訓練模型。

OBJECTIVE str

指定演算法辨識的目標。Amazon Redshift 可支援 reg:squarederror、reg:squaredlogerror、reg:logistic、reg:pseudohubererror、reg:tweedie、binary:logisti 如需這些目標的相關資訊，請參閱 XGBoost 文件中的學習任務參數。

HYPERPARAMETERS { DEFAULT | DEFAULT EXCEPT (key 'value' (,..)) }

指定使用預設的 XGBoost 參數或以使用者指定的值覆寫。這些值必須用單引號括住。以下是 XGBoost 及其預設值的參數範例。

參數名稱	參數值	預設值	備註
num_class	Integer	多類別分類所需。	N/A
num_round	Integer	100	N/A
tree_method	字串	Auto	N/A
max_depth	Integer	6	[0 , 10]
min_child_weight	Float	1	MinValue: 0, MaxValue
子範例	Float	1	MinValue MaxValue：半點
Gamma	Float	0	MinValue: 0, MaxValue: 5
alpha	Float	0	MinValue: 0, MaxValue
eta	Float	0.3	MinValue：零 MaxValue
colsample_byleve	Float	1	MinValue：零一 MaxValue

參數名稱	參數值	預設值	備註
colsample_bynode	Float	1	MinValue：零一 MaxValue
colsample_bytree	Float	1	MinValue MaxValue：半點
lambda	Float	1	MinValue: 0, MaxValue
max_delta_step	Integer	0	[0, 10]

下列範例會準備 XgBoost 的資料。

```
DROP TABLE IF EXISTS abalone_xgb;

    CREATE TABLE abalone_xgb (
    length_val float,
    diameter float,
    height float,
    whole_weight float,
    shucked_weight float,
    viscera_weight float,
    shell_weight float,
    rings int,
    record_number int);

    COPY abalone_xgb
    FROM 's3://redshift-downloads/redshift-ml/abalone_xg/'
    REGION 'us-east-1'
    IAM_ROLE default
    IGNOREHEADER 1 CSV;
```

下列範例會使用指定的進階選項來建立 XGBoost 模型，例如 MODEL_TYPE、OBJECTIVE 和 PREPROCESSORS。

```
DROP MODEL abalone_xgboost_multi_predict_age;
```

```
    CREATE MODEL abalone_xgboost_multi_predict_age
    FROM ( SELECT length_val,
                  diameter,
                  height,
                  whole_weight,
                  shucked_weight,
                  viscera_weight,
                  shell_weight,
                  rings
   FROM abalone_xgb WHERE record_number < 2500 )
    TARGET rings FUNCTION ml_fn_abalone_xgboost_multi_predict_age
    IAM_ROLE default
    AUTO OFF
    MODEL_TYPE XGBOOST
    OBJECTIVE 'multi:softmax'
    PREPROCESSORS 'none'
    HYPERPARAMETERS DEFAULT EXCEPT (NUM_ROUND '100', NUM_CLASS '30')
    SETTINGS (S3_BUCKET 'your-bucket');
```

下列範例會使用推論查詢來對記錄編號大於 2500 的魚預測年齡。其使用從上述命令中建立的函數 ml_fn_abalone_xgboost_multi_predict_age。

```
select ml_fn_abalone_xgboost_multi_predict_age(length_val,
                                                diameter,
                                                height,
                                                whole_weight,
                                                shucked_weight,
                                                viscera_weight,
                                                shell_weight)+1.5 as age
from abalone_xgb where record_number > 2500;
```

使用自有模型 (BYOM) - 本機推論

Amazon Redshift ML 支援透過使用自有模型 (BYOM) 進行本機推論。

以下概述 BYOM 的 CREATE MODEL 語法基本選項。您可以使用在 Amazon Redshift 以外訓練的模型與 Amazon 一起在亞馬 Amazon SageMaker Redshift 本地進行資料庫內推論。

用於本機推論的 CREATE MODEL 語法

以下說明用於本機推論的 CREATE MODEL 語法。

```
CREATE MODEL model_name
```

```
    FROM ('job_name' | 's3_path' )
    FUNCTION function_name ( data_type [, ...] )
    RETURNS data_type
    IAM_ROLE { default }
    [ SETTINGS (
      S3_BUCKET 'bucket', | --required
      KMS_KEY_ID 'kms_string') --optional
    ];
```

Amazon Redshift 目前僅支援對 BYOM 使用預先訓練的 XGBoost、MLP 和線性學習程式模型。您可以使用此路徑匯入直接在 Amazon SageMaker 訓練的 SageMaker Autopilot 自動輔助駕駛和模型，以進行本地推論。

用於本機推論的 CREATE MODEL 參數

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

FROM ('job_name' | 's3_path')

該 job_name 使用 Amazon 任 SageMaker 務名稱作為輸入。任務名稱可以是 Amazon SageMaker 培訓任務名稱，也可以是 Amazon SageMaker 自動駕駛儀任務名稱。該任務必須在擁有 Amazon Redshift 叢集的相同 AWS 帳戶中建立。要查找任務名稱，請啟動 Amazon SageMaker。在訓練下拉式選單中，選擇訓練工作。

's3_path' 會指定 .tar.gz 模型成品檔案的 S3 位置，這會在建立模型時使用。

FUNCTION function_name (data_type [, ...])

要建立的函數名稱和輸入引數的資料類型。您可以提供結構描述名稱。

RETURNS data_type

函數所傳回值的資料類型。

IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>'}

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREATE MODEL 命令時與叢集關聯的 IAM 角色。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。

SETTINGS (S3_BUCKET 'bucket', | KMS_KEY_ID 'kms_string')

S3_BUCKET 子句會指定用來儲存中繼結果的 Amazon S3 位置。

(選擇性) KMS_KEY_ID 子句指定 Amazon Redshift 是否使用伺服器端加密搭配 AWS KMS 金鑰來保護靜態資料。傳輸中的資料會受到 Secure Sockets Layer (SSL) 保護。

如需詳細資訊，請參閱 CREATE MODEL 和使用者指引。

用於本機推論的 CREATE MODEL 範例

下列範例會建立先前在 Amazon 亞馬 Amazon SageMaker Redshift 以外接受過訓練的模型。由於 Amazon Redshift ML 支援該模型類型來進行本機推論，因此下列 CREATE MODEL 會建立可在 Amazon Redshift 本端使用的函數。您可以提供 SageMaker 訓練工作名稱。

```
CREATE MODEL customer_churn
    FROM 'training-job-customer-churn-v4'
    FUNCTION customer_churn_predict (varchar, int, float, float)
    RETURNS int
    IAM_ROLE default
    SETTINGS (S3_BUCKET 'your-bucket');
```

建立模型之後，您可以使用函數 customer_churn_predict 與指定的引數類型來進行預測。

使用自有模型 (BYOM) - 遠端推論

Amazon Redshift ML 也支援透過使用自有模型 (BYOM) 進行遠端推論。

以下概述 BYOM 的 CREATE MODEL 語法基本選項。

⚠ 這是超級資料類型的預先發行文件，可在預覽版中輸入至 Amazon Redshift ML 模型的 BYOM 模型。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

指定將 SUPER 資料類型用作輸入資料，而傳回的資料類型表示您要建立在 Amazon SageMaker JumpStart 中託管的大型語言模型 (LLM)。建立 LLM 目前僅可做為預覽功能。此預覽可在下列中使用 AWS 區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)

- 歐洲 (斯德哥爾摩) (eu-north-1)

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。 AWS 區域 會列出目前帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的建立叢集。
5. 選擇建立叢集按鈕以建立預覽叢集。

   > **Note**
   >
   > `preview_2023` 軌跡是最近可用的預覽軌跡。此軌跡僅支援使用 RA3 節點類型建立叢集。不支援節點類型 DC2 和任何較舊的節點類型。

6. 當您的預覽叢集可用時，請使用 SQL 用戶端載入和查詢資料。

您也可以建立預覽工作群組以建立 LLM。您無法在生產環境中使用這些功能，也無法將工作群組移至另一個工作群組。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。如需如何建立預覽工作群組的指示，請參閱建立預覽工作群組。

用於遠端推論的 CREATE MODEL 語法

以下說明用於遠端推論的 CREATE MODEL 語法。

```
CREATE MODEL model_name
    FUNCTION function_name ( data_type [, ...] )
    RETURNS data_type
    SAGEMAKER 'endpoint_name'['model_name']
    IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
```

用於遠端推論的 CREATE MODEL 參數

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

FUNCTION fn_name ([data_type] [, ...])

函數的名稱和輸入引數的資料類型。請參閱所有支援資料類型的資料類型。Geography、geometry 和 hllsketch 不受支援。指定將 SUPER 資料類型用作輸入資料，而傳回的資料類型表示您要建立在 Amazon SageMaker JumpStart 中託管的大型語言模型 (LLM)。

或者，您也可以指定只使用 SUPER 資料類型做為輸入資料，而不用作傳回的資料類型。使用 SUPER 資料類型作為輸入僅可作為預覽功能使用。

您也可以提供結構描述名稱，而不是函數名稱。

RETURNS data_type

函數所傳回值的資料類型。請參閱所有支援資料類型的資料類型。Geography、geometry 和 hllsketch 不受支援。指定將 SUPER 資料類型用作輸入資料，而傳回的資料類型表示您要建立在 Amazon SageMaker JumpStart 中託管的大型語言模型 (LLM)。

或者，您也可以指定只使用 SUPER 資料類型做為傳回的資料類型，而不用作輸入資料。

SAGEMAKER 'endpoint_name'[:'model_name']

Amazon SageMaker 端點的名稱。如果端點名稱指向多模型端點，請新增要使用的模型名稱。端點必須託管在與 Amazon Redshift 叢集相同的 AWS 區域中。若要尋找您的端點，請啟動 Amazon SageMaker。在推論下拉式功能表中，選擇端點。

IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>'}

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREATE MODEL 命令時與叢集關聯的 IAM 角色。或者，您也可以指定 IAM 角色的 ARN 以使用該角色。

將模型部署到 SageMaker 端點時， SageMaker 會在 Amazon Redshift 中建立模型的資訊。然後透過外部函數執行推斷。您可以使用 SHOW MODEL 命令來檢視 Amazon Redshift 叢集上的模型資訊。

用於遠端推論的 CREATE MODEL 使用須知

使用 CREATE Model 進行遠端推論之前，請考慮下列事項：

- 如果您使用 SUPER 資料類型作為輸入資料，且傳回的輸出也必須是 SUPER 資料類型，BYOM 模型只能支援一個引數。

- 模型必須接受以逗號分隔值 (CSV) 格式的輸入，透過中的文字 /CSV 內容類型。 SageMaker僅當您未使用 SUPER 資料類型當做輸入時才適用。
- 端點必須由擁有 Amazon Redshift 叢集的相同 AWS 帳戶託管。
- 模型的輸出必須是建立函數時指定類型的單一值，其格式為逗號分隔值 (CSV) 至中的文字 /CSV 內容類型。 SageMakerVarchar 資料類型不能在引號中，每個輸出都必須在新的一行中。僅當您指定模型不應該傳回 SUPER 資料類型時才適用。
- 模型接受 null 作為空字串。
- 請確定 Amazon SageMaker 端點有足夠的資源來容納來自 Amazon Redshift 的推論呼叫，或是可以自動擴展 Amazon SageMaker 端點。
- 當傳回的類型是 SUPER 時，模型輸出必須是 JSON 和 `application/jsonlines` 內容類型。
- 當輸入和輸出類型都是 SUPER 時，模型必須接受並透過內容類型 `application/json` 傳回 JSON。

用於遠端推論的 CREATE MODEL 範例

下列範例會建立使用 SageMaker 端點進行預測的模型。請確定端點正在執行以進行預測，並在 CREATE MODEL 命令中指定其名稱。

```
CREATE MODEL remote_customer_churn
    FUNCTION remote_fn_customer_churn_predict (varchar, int, float, float)
    RETURNS int
    SAGEMAKER 'customer-churn-endpoint'
    IAM_ROLE default;
```

下列範例透過使用 SUPER 資料類型做為輸入資料，建立大型語言模型模型 (LLM)，並輸出 SUPER 資料類型。有限責任公司託管在 SageMaker 快速啟動。

```
CREATE MODEL sample_super_data_model
FUNCTION sample_super_data_model_predict(super)
RETURNS super
SAGEMAKER 'sample_super_data_model_endpoint'
IAM_ROLE default;
```

CREATE MODEL 與 K-MEANS

Amazon Redshift 支援可將未標示資料分組的 K 平均值演算法。此演算法可解決您想在資料中探索分組的叢集問題。未分類的資料會根據其相似性和差異進行分組和分割。

CREATE MODEL 與 K-MEANS 語法

```
CREATE MODEL model_name
    FROM { table_name | ( select_statement ) }
    FUNCTION function_name
    IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
    AUTO OFF
    MODEL_TYPE KMEANS
    PREPROCESSORS 'string'
    HYPERPARAMETERS DEFAULT EXCEPT ( K 'val' [, ...] )
    SETTINGS (
      S3_BUCKET 'bucket',
      KMS_KEY_ID 'kms_string', |
        -- optional
      S3_GARBAGE_COLLECT on / off, |
        -- optional
      MAX_CELLS integer, |
        -- optional
      MAX_RUNTIME integer
        -- optional);
```

CREATE MODEL 與 K-MEANS 參數

AUTO OFF

關閉預處理器、演算法和超參數選擇的 CREATE MODEL 自動探索。

MODEL_TYPE KMEANS

指定使用 KMEANS 來訓練模型。

PREPROCESSORS 'string'

指定特定資料欄集的特定預處理器組合。格式會是 columnSets 清單，以及要套用到每一組資料欄的適當轉換。Amazon Redshift 支持 3 K-均值預處理器，即 StandardScaler， MinMax和。NumericPassthrough如果您不想對 K-Means 應用任何預處理，請 NumericPassthrough 明確選擇作為變壓器。如需轉換器的相關資訊，請參閱 CREATE MODEL 和使用者指引參數。

K 平均值演算法會使用歐幾里德距離來計算相似性。預處理資料可確保模型的特徵保持相同的比例並產生可靠的結果。

HYPERPARAMETERS DEFAULT EXCEPT (K 'val' [, ...])

指定是否使用 K 平均值參數。使用 K 平均值演算法時，您必須指定 K 參數。如需詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的 K-Means 超參數

下列範例會準備 K 平均值的資料。

```
CREATE MODEL customers_clusters
FROM customers
FUNCTION customers_cluster
IAM_ROLE default
AUTO OFF
MODEL_TYPE KMEANS
PREPROCESSORS '[
{
  "ColumnSet": [ "*" ],
  "Transformers": [ "NumericPassthrough" ]
}
]'
HYPERPARAMETERS DEFAULT EXCEPT ( K '5' )
SETTINGS (S3_BUCKET 'bucket');

select customer_id, customers_cluster(...) from customers;
customer_id | customers_cluster
--------------------
12345            1
12346            2
12347            4
12348            0
```

CREATE MODEL 與預測

Redshift ML 中的預測模型會使用 Amazon Forecast 來建立準確的時間序列預測。這樣做可讓您使用一段時間內的歷史資料來預測未來事件。Amazon Forecast 的常見使用案例包括使用零售產品資料來決定如何為庫存定價格、製造數量資料來預測要訂購多少項目，以及使用 Web 流量資料來預測 Web 伺服器可能接收的流量。

Amazon Forecast 的配額限制會在 Amazon Redshift 預測模型中強制執行。例如，預測的最大數量為 100，但可調整。捨棄預測模型不會自動刪除 Amazon Forecast 中的相關聯資源。如果您刪除 Redshift 叢集，所有相關聯的模型也會被刪除。

請注意，預測模型目前僅適用於下列區域：

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (奧勒岡) (us-west-2)

- 亞太區域 (孟買) (ap-south-1)
- 亞太區域 (首爾) (ap-northeast-2)
- 亞太區域 (新加坡) (ap-southeast-1)
- 亞太區域 (雪梨) (ap-southeast-2)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (法蘭克福) (eu-central-1)
- 歐洲 (愛爾蘭) (eu-west-1)

CREATE MODEL 與預測語法

```
CREATE [ OR REPLACE ] MODEL forecast_model_name
FROM { table_name | ( select_query ) }
TARGET column_name
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>'}
AUTO ON
MODEL_TYPE FORECAST
SETTINGS (
  S3_BUCKET 'bucket',
  HORIZON integer,
  FREQUENCY forecast_frequency
  [PERCENTILES '0.1', '0.5', '0.9']
```

CREATE MODEL 與預測參數

forecast_model_name

模型的名稱。模型名稱必須是唯一的。

FROM { table_name | (select_query) }

指定訓練資料的 table_name 或查詢。這可以是系統中現有的資料表，也可以是與 Amazon RedShift 相容的 SELECT 查詢 (以括號括住)。資料表或查詢結果至少必須有三個資料欄：(1) 指定時間序列名稱的 varchar 資料欄。每個資料集可以有多個時間序列；(2) 一個日期時間資料欄；以及 (3) 要預測的目標資料欄。此目標資料欄必須是整數或浮點數。如果您提供的資料集包含三個以上的資料欄，Amazon Redshift 會假設所有其他資料欄都是相關時間序列的一部分。請注意，相關的時間序列必須是整數或浮點數類型。如需相關時間序列的相關資訊，請參閱使用相關時間序列資料集。

TARGET column_name

成為預測目標的資料欄名稱。該資料欄必須存在於 FROM 子句中。

IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 CREAT MODEL 命令時與叢集關聯的 IAM 角色。或者，您可以指定 IAM 角色的 ARN 以使用該角色。

AUTO ON

開啟演算法和超參數選擇的 CREATE MODEL 自動探索。在建立 Forecast 模型時指定開啟表示使用 Forecast AutoPredictor，其中 Amazon Forecast 會將最佳演算法組合套用至資料集中的每個時間序列。

MODEL_TYPE FORECAST

指定使用 FORECAST 來訓練模型。

S3_BUCKET 'bucket'

您之前建立的 Amazon Simple Storage Service 儲存貯體名稱，用於在 Amazon Redshift 和 Amazon Forecast 之間共用訓練資料和成品。Amazon Redshift 會在卸載訓練資料之前，在此儲存貯體中建立一個子資料夾。訓練完成後，Amazon Redshift 會刪除建立的子資料夾及其內容。

HORIZON 整數

預測模型可傳回的預測數量上限。模型一經過訓練，就無法再變更此整數。

FREQUENCY forecast_frequency

指定您想要的預測細微程度。可用選項為 `Y | M | W | D | H | 30min | 15min | 10min | 5min | 1min`。如果您正在訓練預測模型，則需要此選項。

PERCENTILES 字串

逗號分隔的字串，可指定用來訓練預測器的預測類型。預測類型可以是 0.01 到 0.99 之間的分位數，增量單位為 0.01 或更高。您也可以使用平均值指定平均預測。您最多可以指定五種預測類型。

下列範例說明如何建立簡單預測模型。

```
CREATE MODEL forecast_example
FROM forecast_electricity_
TARGET target
IAM_ROLE 'arn:aws:iam::<account-id>:role/<role-name>'
AUTO ON
```

```
MODEL_TYPE FORECAST
SETTINGS (S3_BUCKET 'redshift-ml-bucket',
          HORIZON 24,
          FREQUENCY 'H',
          PERCENTILES '0.25,0.50,0.75,mean',
          S3_GARBAGE_COLLECT OFF);
```

建立預測模型後，您可以使用預測資料建立新資料表。

```
CREATE TABLE forecast_model_results as SELECT Forecast(forecast_example)
```

然後，您可以查詢新資料表以取得預測。

```
SELECT * FROM forecast_model_results
```

CREATE PROCEDURE

為目前的資料庫建立新的預存程序或取代現有的程序。

如需詳細資訊和範例，請參閱 在 Amazon Redshift 中建立預存程序。

所需權限

您必須具有下列其中一種方式的權限，才能執行「建立或取代程序」：

- 對於 CREATE PROCEDURE：
  - 超級使用者
  - 在建立預存程序的結構描述上具有 CREATE 和 USING 權限的使用者
- 對於 REPLACE PROCEDURE：
  - 超級使用者
  - 程序擁有者

語法

```
CREATE [ OR REPLACE ] PROCEDURE sp_procedure_name
  ( [ [ argname ] [ argmode ] argtype [, ...] ] )
[ NONATOMIC ]
AS $$
  procedure_body
```

```
$$ LANGUAGE plpgsql
[ { SECURITY INVOKER | SECURITY DEFINER } ]
[ SET configuration_parameter { TO value | = value } ]
```

參數

OR REPLACE

此子句指定如果已有一個程序的名稱和輸入引數資料類型 (或簽章) 與此程序相同，則取代現有程序。您只能將程序取代為定義一組相同資料類型的新程序。

如果您定義的程序與現有程序同名，但簽章不同，則會建立新程序。換言之，程序名稱是過載。如需詳細資訊，請參閱 多載程序名稱。

sp_procedure_name

程序的名稱。如果您指定結構描述名稱 (例如 **myschema.myprocedure**)，則會在指定的結構描述中建立程序。否則會在目前結構描述中建立程序。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

我們建議您在所有預存程序名稱前加上 sp_。Amazon Redshift 會保留預存程序名稱的 sp_ 字首。您可以使用 sp_ 字首，以確保您的預存程序不會與任何現有或未來的 Amazon Redshift 內建預存程序或函數名稱發生衝突。如需詳細資訊，請參閱 命名預存程序。

如果輸入引數的資料類型 (或簽章) 不同，您可以定義多個同名的程序。換言之，在此情況下程序名稱是過載。如需更多資訊，請參閱多載程序名稱

[argname] [argmode] argtype

引數名稱、引數模式和資料類型的清單。僅資料類型是必要的。名稱和模式是選用的，其位置可調換。

引數模式可以是 IN、OUT 或 INOUT。預設值為 IN。

您可以使用 OUT 和 INOUT 引數，從程序呼叫傳回一或多個值。有 OUT 或 INOUT 引數時，程序呼叫會傳回一個包含 n 欄的結果列，其中 n 是 OUT 或 INOUT 引數總數。

INOUT 引數同時為輸入和輸出引數。輸入引數包括 IN 和 INOUT 引數，輸出引數包括 OUT 和 INOUT 引數。

CALL 陳述式中不指定 OUT 引數。在預存程序 CALL 陳述式中，請指定 INOUT 引數。從巢狀呼叫傳遞和傳回值，以及傳回 refcursor 時，INOUT 引數可能很有用。如需 refcursor 類型的詳細資訊，請參閱游標。

引數資料類型可以是任何標準 Amazon Redshift 資料類型。此外，引數資料類型還可以是 `refcursor`。

您最多可以指定 32 個輸入引數和 32 個輸出引數。

AS $$ procedure_body $$

包圍要執行之程序的結構。常值關鍵字 AS $$ 和 $$ 是必要的。

Amazon Redshift 會要求您使用稱為 $ 符號引用的格式包圍程序中的陳述式。包圍範圍當中的任何內容都會原封不動傳遞。您不需要逸出任何特殊字元，因為字串的內容是逐字撰寫。

使用 $ 符號引用時，您會使用一組 $$ 符號配對表示要執行之陳述式的開頭和結尾，如以下範例所示。

```
$$ my statement $$
```

您也可以選擇在每組配對的兩個 $ 符號之間指定字串來協助識別陳述式。您在包圍配對的開頭和結尾中使用的字串必須相同。此字串區分大小寫，且遵循與未加引號的識別碼相同的限制條件，不過後者不可包含 $ 符號。下列範例使用字串 test。

```
$test$ my statement $test$
```

此語法也適用於巢狀 $ 符號引用。如需 $ 符號引用的詳細資訊，請參閱 PostgreSQL 文件中的辭典結構下的「$ 符號引用的字串常數」。

procedure_body

一組有效的 PL/pgSQL 陳述式。PL/pgSQL 陳述式以程序性建構 (包括迴圈和條件式表達式) 來擴增 SQL 命令，以控制邏輯流程。大部分 SQL 命令可用於程序主體中，包括資料修改語言 (DML) (例如 COPY、UNLOAD 和 INSERT) 和資料定義語言 (DDL) (例如 CREATE TABLE)。如需詳細資訊，請參閱 PL/pgSQL 語言參考。

LANGUAGE plpgsql

語言值。指定 `plpgsql`。您必須具有語言的使用許可，才能使用 `plpgsql`。如需詳細資訊，請參閱 GRANT。

NONATOMIC

在非原子交易模式中建立預存程序。NONATOMIC 模式會自動提交程序中的陳述式。此外，當 NONATOMIC 程序內部發生錯誤時，如果錯誤是由例外狀況區塊處理，則不會重新擲回錯誤。如需詳細資訊，請參閱 管理交易 及 RAISE。

將預存程序定義為 NONATOMIC 時，請考慮下列事項：

- 當您巢狀化預存程序呼叫時，必須以相同的交易模式建立所有程序。
- 在 NONATOMIC 模式下建立程序時，不支援 SECURITY DEFINER 選項和 SET configuration_parameter 選項。
- 處理隱含遞交時，任何已開啟的 (明確或隱含) 游標皆會自動關閉。因此，您必須在開始游標循環之前開啟明確的交易，以確保循環迭代中的任何 SQL 都不會隱含遞交。

SECURITY INVOKER | SECURITY DEFINER

指定 NONATOMIC 時不支援 SECURITY DEFINER 選項。

程序的安全模式決定程序在執行時間的存取權限。程序必須有存取基礎資料庫物件的許可。

若為 SECURITY INVOKER 模式，程序會使用呼叫程序的使用者的權限。使用者必須具有基礎資料庫物件的明確權限。預設為 SECURITY INVOKER。

對於 SECURITY DEFINER 模式，程序會使用程序擁有者的權限。程序擁有者定義為在執行階段擁有該程序的使用者，而不一定是最初定義程序的使用者。呼叫程序的使用者必須有程序的執行權限，但不需要基礎物件的任何權限。

SET configuration_parameter { TO value | = value }

指定 NONATOMIC 時不支援這些選項。

SET 子句會在進入程序時將指定的 configuration_parameter 設為指定的值。當程序離開時，此子句會接著將 configuration_parameter 還原到先前的值。

使用須知

如果使用 SECURITY DEFINER 選項建立預存程序，則在從預存程序中調用 CURRENT_USER 函數時，Amazon Redshift 會傳回預存程序擁有者的使用者名稱。

範例

> **Note**
>
> 如果在執行這些範例時，您遇到類似以下的錯誤：
>
> ```
> ERROR: 42601: [Amazon](500310) unterminated dollar-quoted string at or near "$$
> ```

請參閱Amazon Redshift 中的預存程序概觀。

下列範例建立具有兩個輸入參數的程序。

```
CREATE OR REPLACE PROCEDURE test_sp1(f1 int, f2 varchar(20))
AS $$
DECLARE
  min_val int;
BEGIN
  DROP TABLE IF EXISTS tmp_tbl;
  CREATE TEMP TABLE tmp_tbl(id int);
  INSERT INTO tmp_tbl values (f1),(10001),(10002);
  SELECT INTO min_val MIN(id) FROM tmp_tbl;
  RAISE INFO 'min_val = %, f2 = %', min_val, f2;
END;
$$ LANGUAGE plpgsql;
```

Note

當您撰寫預存程序時，我們建議您採用保護敏感值的最佳作法：
不要在預存程序邏輯中對任何敏感資訊進行硬式編碼。例如，請勿在預存程序主體的 CREATE USER 陳述式中指派使用者密碼。這會造成安全性風險，因為硬式編碼值可以記錄為目錄資料表中的結構描述資料。應改為透過參數將敏感值 (例如密碼) 當做引數傳遞給預存程序。
如需預存程序的相關資訊，請參閱 CREATE PROCEDURE 和在 Amazon Redshift 中建立預存程序。如需目錄資料表的相關資訊，請參閱系統目錄資料表。

下列範例建立具有一個 IN 參數、一個 OUT 參數和一個 INOUT 參數的程序。

```
CREATE OR REPLACE PROCEDURE test_sp2(f1 IN int, f2 INOUT varchar(256), out_var OUT
 varchar(256))
AS $$
DECLARE
  loop_var int;
BEGIN
  IF f1 is null OR f2 is null THEN
    RAISE EXCEPTION 'input cannot be null';
  END IF;
  DROP TABLE if exists my_etl;
```

```
  CREATE TEMP TABLE my_etl(a int, b varchar);
    FOR loop_var IN 1..f1 LOOP
        insert into my_etl values (loop_var, f2);
        f2 := f2 || '+' || f2;
    END LOOP;
  SELECT INTO out_var count(*) from my_etl;
END;
$$ LANGUAGE plpgsql;
```

CREATE RLS POLICY

建立新的資料列層級安全性政策，以提供資料庫物件的精細存取權。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以建立政策。

語法

```
CREATE RLS POLICY policy_name
[ WITH (column_name data_type [, ...]) [ [AS] relation_alias ] ]
USING ( using_predicate_exp )
```

參數

policy_name

政策的名稱。

WITH (column_name data_type [, ...])

指定參考政策附加資料表資料欄的 column_name 和 data_type。

只有當 RLS 政策未參考任何附加政策的資料表資料欄時，您才可以省略 WITH 子句。

AS relation_alias

為要附加 RLS 政策的資料表指定選擇性別名。

USING (using_predicate_exp)

指定套用至查詢中 WHERE 子句的篩選條件。Amazon Redshift 會在查詢層級使用者述詞之前套用政策述詞。例如，**current_user = 'joe' and price > 10** 會限制 Joe 只能查看價格大於 $10 的記錄。

使用須知

使用 CREATE RLS POLICY 陳述式時，請注意以下內容：

- Amazon Redshift 支援篩選，這些篩選可以是查詢中 WHERE 子句的一部分。
- 所有附加至資料表的政策都必須使用相同的資料表別名建立。
- 您不需要查閱資料表上的 SELECT 權限。當您建立政策時，Amazon Redshift 會針對個別政策授予查閱資料表上的 SELECT 權限。查閱資料表是在政策定義中使用的資料表物件。
- Amazon Redshift 資料列層級安全性不支援在政策定義中使用下列物件類型：目錄資料表、跨資料庫關聯性、外部資料表、一般檢視、後期繫結檢視、開啟 RLS 政策的資料表和暫存資料表。

範例

下列 SQL 陳述式會建立用於 CREATE RLS POLICY 範例的資料表、使用者和角色。

```
-- Create users and roles reference in the policy statements.
CREATE ROLE analyst;

CREATE ROLE consumer;

CREATE USER bob WITH PASSWORD 'Name_is_bob_1';

CREATE USER alice WITH PASSWORD 'Name_is_alice_1';

CREATE USER joe WITH PASSWORD 'Name_is_joe_1';

GRANT ROLE sys:secadmin TO bob;

GRANT ROLE analyst TO alice;

GRANT ROLE consumer TO joe;

GRANT ALL ON TABLE tickit_category_redshift TO PUBLIC;
```

以下範例會建立稱為 policy_concerts 的政策。

```
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');
```

CREATE ROLE

建立作為權限集合的新自訂角色。如需 Amazon Redshift 系統定義角色的清單，請參閱 the section called “Amazon Redshift 系統定義角色”。查詢 SVV_ROLES 以檢視叢集或工作群組中目前建立的角色。

可以建立的角色數量有配額限制。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制。

所需的許可

以下是 CREATE ROLE 所需的權限。

- 超級使用者
- 具有 CREATE ROLE 權限的使用者

語法

```
CREATE ROLE role_name
[ EXTERNALID external_id ]
```

參數

role_name

角色的名稱。角色名稱必須是唯一的，且不能與任何使用者名稱相同。角色名稱不能是保留字。

超級使用者或擁有 CREATE ROLE 權限的一般使用者可以建立角色。如果使用者不是超級使用者，但使用者已被授予 WITH GRANT OPTION 角色和 ALTER 權限的 USAGE 權限，就可以將此角色授予任何人。

EXTERNALID external_id

與身分提供者相關聯的角色識別碼。如需詳細資訊，請參閱 Amazon Redshift 的原生身分提供者 (IdP) 聯合合。

範例

下列範例會建立 `sample_role1` 角色。

```
CREATE ROLE sample_role1;
```

下列範例會使用與身分提供者相關聯的外部 ID 來建立角色 sample_role1。

```
CREATE ROLE sample_role1 EXTERNALID "ABC123";
```

CREATE SCHEMA

為目前資料庫定義新的結構描述。

所需權限

以下是 CREATE SCHEMA 所需的權限：

- 超級使用者
- 具有 CREATE SCHEMA 權限的使用者

語法

```
CREATE SCHEMA [ IF NOT EXISTS ] schema_name [ AUTHORIZATION username ]
           [ QUOTA {quota [MB | GB | TB] | UNLIMITED} ] [ schema_element [ ... ]

CREATE SCHEMA AUTHORIZATION username[ QUOTA {quota [MB | GB | TB] | UNLIMITED} ]
 [ schema_element [ ... ] ]
```

參數

IF NOT EXISTS

此子句會指出，若指定的結構描述已存在，則命令不應進行任何變更，且應傳回結構描述存在的訊息，而不是在發生錯誤的情況下終止。

此子句在編寫指令碼時很實用，如此指令碼就不會因為 CREATE SCHEMA 嘗試建立已存在的結構描述而失敗。

schema_name

新結構描述的名稱。結構描述名稱不可以是 PUBLIC。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

> **Note**
>
> search_path 組態參數中結構描述的清單會決定，在沒有結構描述名稱的情況下參考名稱相同的物件時，這些物件的優先順序。

AUTHORIZATION

將所有權提供給指定使用者的子句。

username

結構描述擁有者的名稱。

schema_element

要在結構描述內建立的一個或多個物件的定義。

QUOTA

所指定之結構描述可以使用的磁碟空間數上限。這個空間是集體磁碟使用量。它包含所有永久資料表、所指定之結構描述下的具體化檢視，以及在每個計算節點上具有 ALL 分佈之所有資料表的複本。結構描述配額不會將暫時資料表納入考量，此暫時資料表是作為暫時命名空間或結構描述一部分而建立的。

若要檢視設定的結構描述配額，請參閱 SVV_SCHEMA_QUOTA_STATE。

若要檢視超出結構描述配額的記錄，請參閱 STL_SCHEMA_QUOTA_VIOLATIONS。

Amazon Redshift 會將選取的值轉換為 MB。GB 是您未指定值時的預設測量單位。

您必須是資料庫超級使用者，才能設定和變更結構描述配額。不是超級使用者，但具有 CREATE SCHEMA 許可的使用者可以建立具有已定義配額的結構描述。當您在未定義配額的情況下建立結構描述時，結構描述會有無限的配額。當您將配額設為低於結構描述所使用的目前值時，除非您釋放磁碟空間，否則 Amazon Redshift 不允許進一步擷取。DELETE 陳述式會從資料表刪除資料，並只在 VACUUM 執行時，才會釋出磁碟空間。

Amazon Redshift 會在確認交易之前檢查每筆交易是否存在配額違規情況。Amazon Redshift 會根據設定的配額，檢查每個已修改結構描述的大小 (結構描述中所有資料表使用的磁碟空間)。因為配額違規檢查發生在交易結束時，所以在確定之前，大小限制可以暫時超過交易內的配額。當交易超過配額時，Amazon Redshift 會停止交易、禁止後續導入，並還原所有變更，直到您釋放磁碟空間

為止。由於背景 VACUUM 和內部清除，有可能在交易取消之後您檢查結構描述時結構描述不是完整的。

在例外情況下，Amazon Redshift 會忽略配額違規，並在特定情況下遞交交易。Amazon Redshift 會針對僅由下列一或多個陳述式組成的交易執行此操作，而且在相同交易中沒有 INSERT 或 COPY 擷取陳述式：

- DELETE
- TRUNCATE
- VACUUM
- DROP TABLE
- 只在將資料從完整結構描述移到另一個非完整結構描述時，ALTER TABLE APPEND 才適用

UNLIMITED

Amazon Redshift 不會限制結構描述總大小的增長。

限制

Amazon Redshift 會對結構描述強制實施下列限制。

- 每個資料庫最多 9900 個結構描述。

範例

下列範例會建立名為 US_SALES 的結構描述，並將所有權提供給使用者 DWUSER。

```
create schema us_sales authorization dwuser;
```

下列範例會建立名為 US_SALES 的結構描述、將擁有權提供給使用者 DWUSER，並將配額設定為 50 GB。

```
create schema us_sales authorization dwuser QUOTA 50 GB;
```

若要檢視新的結構描述，請查詢 PG_NAMESPACE 目錄資料表，如下所示。

```
select nspname as schema, usename as owner
```

```
from pg_namespace, pg_user
where pg_namespace.nspowner = pg_user.usesysid
and pg_user.usename ='dwuser';

   schema |  owner
----------+----------
 us_sales | dwuser
(1 row)
```

下列範例會建立 US_SALES 結構描述，或不執行任何動作，並於該結構描述已存在時傳回訊息。

```
create schema if not exists us_sales;
```

CREATE TABLE

在目前資料庫中建立新的資料表。您可以定義資料欄清單，每個資料欄都會保留不同類型的資料。資料表的擁有者是 CREATE TABLE 命令的發行者。

所需權限

以下是 CREATE TABLE 所需的權限：

- 超級使用者
- 具有 CREATE TABLE 權限的使用者

語法

```
CREATE [ [LOCAL ] { TEMPORARY | TEMP } ] TABLE
[ IF NOT EXISTS ] table_name
( { column_name data_type [column_attributes] [ column_constraints ]
  | table_constraints
  | LIKE parent_table [ { INCLUDING | EXCLUDING } DEFAULTS ] }
  [, ... ]  )
[ BACKUP { YES | NO } ]
[table_attributes]

where column_attributes are:
  [ DEFAULT default_expr ]
  [ IDENTITY ( seed, step ) ]
  [ GENERATED BY DEFAULT AS IDENTITY ( seed, step ) ]
```

```
  [ ENCODE encoding ]
  [ DISTKEY ]
  [ SORTKEY ]
  [ COLLATE CASE_SENSITIVE | COLLATE CASE_INSENSITIVE  ]

and column_constraints are:
  [ { NOT NULL | NULL } ]
  [ { UNIQUE  |  PRIMARY KEY } ]
  [ REFERENCES reftable [ ( refcolumn ) ] ]

and table_constraints  are:
  [ UNIQUE ( column_name [, ... ] ) ]
  [ PRIMARY KEY ( column_name [, ... ] )  ]
  [ FOREIGN KEY (column_name [, ... ] ) REFERENCES reftable [ ( refcolumn ) ]


and table_attributes are:
  [ DISTSTYLE { AUTO | EVEN | KEY | ALL } ]
  [ DISTKEY ( column_name ) ]
  [ [COMPOUND | INTERLEAVED ] SORTKEY ( column_name [,...]) |  [ SORTKEY AUTO ] ]
  [ ENCODE AUTO ]
```

參數

LOCAL

選用。雖然陳述式中可接受此關鍵字，但是在 Amazon Redshift 中沒有作用。

TEMPORARY | TEMP

建立臨時資料表的關鍵字，只能在目前工作階段中看見。資料表會在建立所在的工作階段結束時自動捨棄。臨時資料表可與永久資料表同名。臨時資料表是以不同的工作階段專屬結構描述建立 (您無法指定此結構描述的名稱)。此臨時結構描述會成為搜尋路徑中的第一個結構描述，因此，除非您以結構描述名稱限定資料表名稱來存取永久資料表，否則臨時資料表的優先順序高於永久資料表。如需結構描述和優先順序的相關資訊，請參閱 search_path。

> **Note**
>
> 根據預設，資料庫使用者依其 PUBLIC 群組中的自動成員資格，具有建立臨時資料表的許可。若要拒絕使用者的此權限，請撤銷 PUBLIC 群組的 TEMP 權限，然後明確將 TEMP 權限僅授予特定使用者或使用者群組。

IF NOT EXISTS

此子句會指出，若指定的資料表已存在，則命令不應進行任何變更，且應傳回資料表存在的訊息，而不是在發生錯誤的情況下停止。請注意，現有資料表可能與這裡建立的資料表完全不一樣；只有資料表名稱進行比較。

此子句在編寫指令碼時很實用，如此指令碼就不會因為 CREATE TABLE 嘗試建立已存在的資料表而失敗。

table_name

要建立的資料表名稱。

> **Important**
>
> 若您指定 '#' 開頭的資料表名稱，所建立的資料表會是臨時資料表。以下是範例：
>
> ```
> create table #newtable (id int);
> ```
>
> 您也可以使用 '# '參考資料表。例如：
>
> ```
> select * from #newtable;
> ```

資料表名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 多位元組字元，最長可達 4 個位元組。Amazon Redshift 會依節點類型強制執行每個叢集的資料表數目配額限制，包括使用者定義的臨時資料表，以及 Amazon Redshift 在查詢處理或系統維護期間建立的臨時資料表。或者，資料表名稱也可透過資料庫和結構描述名稱來限定。在以下範例中，資料庫名稱為 tickit，結構描述名稱為 public，而資料表名稱為 test。

```
create table tickit.public.test (c1 int);
```

如果資料庫或結構描述不存在，則不會建立資料表，而且陳述式會傳回錯誤。您無法在系統資料庫 template0、template1、padb_harvest 或 sys:internal 中建立資料表或視觀表。

若提供結構描述名稱，則會在該結構描述中建立新資料表 (假設建立者具有存取結構描述的權限)。資料表名稱對於該結構描述來說必須是唯一的。如果未指定結構描述，則會使用目前資料庫結構描述建立資料表。如果您要建立臨時資料表，就不能指定結構描述名稱，因為臨時資料表會採用特殊結構描述。

若臨時資料表是在不同的工作階段中建立的話，在同一個資料庫中可同時有多個同名的臨時資料表存在，因為資料表會指派至不同的結構描述。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

column_name

要在新資料表中建立的資料欄名稱。資料欄名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 多位元組字元，最長可達 4 個位元組。單一資料表中可定義的資料欄數目上限為 1,600 個。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

> **Note**
>
> 若您要建立「寬資料表」，則務必注意，在載入和查詢處理期間，您的資料欄清單未超過中繼結果的資料列寬度界限。如需詳細資訊，請參閱 使用須知。

data_type

要建立之資料欄的資料類型。若是 CHAR 和 VARCHAR 資料欄，您可以改用 MAX 關鍵字，而不宣告長度上限。MAX 會將 CHAR 的長度上限設定為 4,096 個位元組，或將 VARCHAR 的長度上限設定為 65535 個位元組。GEOMETRY 物件的大小上限是 1,048,447 位元組。

如需有關 Amazon Redshift 支援資料類型的資訊，請參閱 資料類型。

DEFAULT default_expr

此子句會指派資料欄的預設資料值。default_expr 的資料類型必須符合資料欄的資料類型。DEFAULT 值必須是無變數的表達式。不允許子查詢、目前資料表中其他資料欄的交叉參考，以及使用者定義的功能。

default_expr 表達式是在未指定資料欄值的任何 INSERT 操作中使用。若未指定預設值，則資料欄的預設值為 null。

若在既定資料欄清單上進行的 COPY 操作省略有 DEFAULT 值的資料欄，則 COPY 命令會插入 default_expr 的值。

IDENTITY(seed, step)

此子句會指出資料欄是 IDENTITY 資料欄。IDENTITY 資料欄包含唯一的自動產生值。IDENTITY 資料欄的資料類型必須是 INT 或 BIGINT。

當您使用 `INSERT` 或 `INSERT INTO [tablename] VALUES()` 陳述式新增資料列時，這些值會從指定為 seed 的值開始，並依指定為 step 的數字遞增。

使用 INSERT INTO [tablename] SELECT * FROM 或 COPY 陳述式載入資料表時，會並行載入資料並將其分發至節點片段。為確保身分值是唯一的，Amazon Redshift 會在建立身分值時略過一些值。身分值是唯一的，但順序可能不符合來源檔案中的順序。

GENERATED BY DEFAULT AS IDENTITY(seed, step)

指定資料欄是預設 IDENTITY 資料欄的子句，其可讓您自動將唯一值指派給資料欄。IDENTITY 資料欄的資料類型必須是 INT 或 BIGINT。當您新增沒有值的資料列時，這些值會從指定為 seed 的值開始，並依指定為 step 的數字遞增。如需如何產生值的詳細資訊，請參閱IDENTITY。

此外，在 INSERT、UPDATE 或 COPY 期間，您可以提供沒有 EXCLIIT_IDS 的值。Amazon Redshift 會使用該值插入身分資料欄，而不是使用系統產生的值。此值可以是複本、小於種子的值，或是介於步驟值之間的值。Amazon Redshift 不會檢查資料欄中的值是否是唯一的。提供一值並不會影響下一個系統產生的值。

> **Note**
>
> 如果您需要資料欄中的唯一性，請不要新增複本值。改為新增小於種子或介於步驟值之間的唯一值。

請記住下列有關預設身分資料欄的事項：

- 預設身分資料欄為 NOT NULL。無法插入 NULL。
- 若要將產生的值插入至預設身分資料欄，請使用關鍵字 DEFAULT。

```
INSERT INTO tablename (identity-column-name) VALUES (DEFAULT);
```

- 置換預設身分資料欄的值並不會影響下一個產生的值。
- 您無法利用 ALTER TABLE ADD COLUMN 陳述式來新增預設身分資料欄。
- 您可以利用 ALTER TABLE APPEND 陳述式來附加預設身分資料欄。

ENCODE encoding

資料欄的壓縮編碼。ENCODE AUTO 是資料表的預設值。Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。如果您為資料表中的任何資料欄指定壓縮編碼，資料表就不會再設定為 ENCODE AUTO。Amazon Redshift 不再自動管理資料表中所有資料欄的壓縮編碼。您可以為資料表指定 ENCODE AUTO 選項，讓 Amazon Redshift 自動管理資料表中所有資料欄的壓縮編碼。

Amazon Redshift 會自動將初始壓縮編碼指派給您未指定壓縮編碼的資料欄，如下所示：

- 暫時資料表中的所有資料欄預設都會有指派的 RAW 壓縮。
- 定義為排序索引鍵的資料欄會有指派的 RAW 壓縮。
- 定義為 BOOLEAN、REAL、DOUBLE PRECISION、GEOMETRY 或 GEOGRAPHY 資料類型的資料行會有指派的 RAW 壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP 或 TIMESTAMPTZ 的資料欄會有指派的 AZ64 壓縮。
- 定義為 CHAR、VARCHAR 或 VARBYTE 的資料欄會有指派的 LZO 壓縮。

> **Note**
>
> 若您不想要壓縮資料欄，請明確指定 RAW 編碼。

支援以下 compression encodings (p. 55)：

- AZ64
- BYTEDICT
- DELTA
- DELTA32K
- LZO
- MOSTLY8
- MOSTLY16
- MOSTLY32
- RAW (無壓縮)
- RUNLENGTH
- TEXT255
- TEXT32K
- ZSTD

DISTKEY

此關鍵字會指定資料欄是資料表的分佈索引鍵。資料表中只能有一個資料欄是分佈索引鍵。您可以在資料欄名稱後面使用 DISTKEY 關鍵字，或使用 DISTKEY (column_name) 語法使其成為資料表定義的一部分。兩種方法的效果一樣。如需詳細資訊，請參閱本主題稍後的 DISTSTYLE 參數。

分佈索引鍵資料欄的資料類型可以是：BOOLEAN、REAL、DOUBLE PRECISION、SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP 或 TIMESTAMPTZ、CHAR 或 VARCHAR。

SORTKEY

此關鍵字會指定資料欄是資料表的排序索引鍵。當資料載入資料表時，資料會依指定為排序索引鍵的一個或多個資料欄排序。您可以在資料欄名稱後面使用 SORTKEY 關鍵字指定單欄排序索引鍵，或使用 SORTKEY (column_name [, ...]) 語法指定一個或多個資料欄做為資料表的排序索引鍵資料欄。此語法只會建立複合排序索引鍵。

您最多可為每個資料表定義 400 個 SORTKEY 資料欄。

排序索引鍵資料欄的資料類型可以是：BOOLEAN、REAL、DOUBLE PRECISION、SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP 或 TIMESTAMPTZ、CHAR 或 VARCHAR。

COLLATE CASE_SENSITIVE | COLLATE CASE_INSENSITIVE

指定資料欄上的字串搜尋或比較是 CASE_SENSITIVE 或 CASE_INSENSITIVE 的子句。預設值與資料庫目前的區分大小寫設定相同。

若要尋找資料庫定序資訊，請使用下列命令：

```
SELECT db_collation();

db_collation
----------------
 case_sensitive
(1 row)
```

NOT NULL | NULL

NOT NULL 會指定不允許資料欄包含 null 值。NULL 是預設值，指出資料欄可接受 null 值。IDENTITY 資料欄預設會宣告為 NOT NULL。

UNIQUE

此關鍵字會指定資料欄只能包含唯一值。唯一資料表限制條件的行為與資料欄限制條件的行為相同，但多了可橫跨多個資料欄的額外功能。若要定義唯一資料表限制條件，請使用 UNIQUE (column_name [, ...]) 語法。

> **Important**
>
> 唯一限制條件僅供參考，系統不會強制執行它們。

PRIMARY KEY

此關鍵字會指定資料欄是資料表的主索引鍵。只能使用資料欄定義將一個資料欄定義為主索引鍵。若要定義具有多資料欄主索引鍵的資料表限制條件，請使用 PRIMARY KEY (column_name [, ...]) 語法。

將資料欄識別為主索引鍵即可提供有關結構描述設計的中繼資料。主索引鍵表示，其他資料表可倚賴這組資料欄做為資料列的唯一識別碼。一個資料表中可指定一個主索引鍵，無論是資料欄限制條件或資料表限制條件都可行。主索引鍵限制條件應命名一組資料欄，這組資料欄有別於由為相同資料表所定義的任何唯一限制條件命名的其他資料欄組。

主索引鍵也會定義為 NOT NULL。

> **Important**
>
> 主索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

References reftable [(refcolumn)]

此子句會指定外部索引鍵限制條件，表示資料欄包含的值只能是符合參考資料表中某一資料列之參考資料欄的值。參考資料欄應為參考資料表中唯一或主索引鍵限制條件的資料欄。

> **Important**
>
> 外部索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

LIKE parent_table [{ INCLUDING | EXCLUDING } DEFAULTS]

此子句會指定現有資料表，新資料表會自動從該資料表複製資料欄名稱、資料類型及 NOT NULL 限制條件。新資料表和父資料表是分開的，對父資料表所做的變更不會套用至新資料表。複製資料欄定義的預設表達式只會在指定了 INCLUDING DEFAULTS 時複製。預設行為是執行預設表達式，如此一來，新資料表中所有資料欄的預設值都是 null。

使用 LIKE 選項建立的資料表不會繼承主索引鍵和外部索引鍵限制條件。LIKE 資料表會繼承分佈樣式、排序索引鍵、BACKUP 及 NULL 屬性，但您無法明確設定這些屬性在 CREATE TABLE ... LIKE 陳述式。

BACKUP { YES | NO }

此子句會指定資料表是否應包含在自動化和手動叢集快照中。若是像臨時資料表這類不會包含重要資料的資料表，指定 BACKUP NO 可以在建立快照以及從快照還原時節省處理時間，並減少 Amazon Simple Storage Service 上的儲存空間。BACKUP NO 設定對於將資料自動複寫到叢集內其他節點的操作並無影響，因此指定了 BACKUP NO 的資料表會在節點故障時還原。預設值為 BACKUP YES。

DISTSTYLE { AUTO | EVEN | KEY | ALL }

定義整個資料表的資料分佈樣式的關鍵字。Amazon Redshift 會根據資料表上指定的分佈樣式，將資料表的資料列分散到運算節點。預設值為 AUTO。

您為資料表選取的分佈樣式會影響資料庫的整體效能。如需詳細資訊，請參閱 使用資料分佈樣式。可能的分佈樣式如下：

- AUTO：Amazon Redshift 會根據資料表資料來指派最佳分佈樣式。例如，如果指定 AUTO 分佈樣式，Amazon Redshift 一開始會將 ALL 分佈樣式指派給小型資料表。當資料表變大時，Amazon Redshift 可能會將分佈樣式變更為 KEY，並選擇主索引鍵 (或複合主鍵的資料欄) 作為 DISTKEY。如果資料表變大，並且沒有任何資料欄適合作為 DISTKEY，則 Amazon Redshift 會將分佈樣式變更為 EVEN。分佈樣式的變更會發生在背景中，對使用者查詢的影響最小。

  若要檢視套用至資料表的分佈樣式，請查詢 PG_CLASS 系統目錄資料表。如需詳細資訊，請參閱 檢視分佈樣式。
- EVEN：資料表中的資料會採循環分佈的方式，均勻分配到叢集中的各個節點。資料列 ID 會用來決定分佈，並且將大致相同的資料列數分佈到每個節點。
- KEY：資料是依 DISTKEY 資料欄中的值分佈。當您將聯結資料表的聯結資料欄設定為分佈索引鍵時，兩個資料表的聯結資料列會在運算節點上並存。資料並存時，最佳化工具就能更有效率地執行聯結。若您指定 DISTSTYLE KEY，則必須命名 DISTKEY 資料欄，無論是資料表或做為資料欄定義的一部分皆可。如需詳細資訊，請參閱本主題前段的 DISTKEY 參數。
- ALL：整個資料表的副本分佈至每個節點。此分佈樣式可確保每個節點上都有任何聯結所需的所有資料列，但儲存空間的需求也會倍增，並且會增加資料表的負載和維護次數。ALL 分佈在 KEY 分佈不適用的情況下搭配特定維度資料表使用時，可改善執行時間，但必須在效能提升與維護成本之間進行權衡。

DISTKEY (column_name)

此限制條件會指定要做為資料表分佈索引鍵的資料欄。您可以在資料欄名稱後面使用 DISTKEY 關鍵字，或使用 DISTKEY (column_name) 語法使其成為資料表定義的一部分。兩種方法的效果一樣。如需詳細資訊，請參閱本主題前段的 DISTSTYLE 參數。

[COMPOUND | INTERLEAVED] SORTKEY (column_name [,...]) | [SORTKEY AUTO]

指定資料表的一個或多個排序索引鍵。當資料載入資料表時，資料會依指定為排序索引鍵的資料欄排序。您可以在資料欄名稱後面使用 SORTKEY 關鍵字指定單欄排序索引鍵，或使用 SORTKEY (column_name [, ...]) 語法指定一個或多個資料欄做為資料表的排序索引鍵資料欄。

您也可以選擇指定 COMPOUND 或 INTERLEAVED 排序樣式。如果您使用資料欄指定 SORTKEY，則預設值為 COMPOUND。如需詳細資訊，請參閱 使用排序索引鍵。

若沒有指定任何排序索引鍵選項，則預設值為 AUTO。

您最多可為每個資料表定義 400 個 COMPOUND SORTKEY 資料欄或 8 個 INTERLEAVED SORTKEY 資料欄。

AUTO

指定 Amazon Redshift 會根據資料表資料指派最佳排序索引鍵。例如，如果指定 AUTO 排序索引鍵，Amazon Redshift 一開始就不會為資料表指派任何排序索引鍵。如果 Amazon Redshift 判斷排序索引鍵可以改善查詢效能，那麼 Amazon Redshift 可能會在變更資料表的排序索引鍵。資料表的實際排序是透過自動資料表排序來完成的。如需詳細資訊，請參閱 自動資料表排序。

Amazon Redshift 不會修改已具有排序或分佈索引鍵的資料表。有一個例外情況是，如果資料表具有從未在 JOIN 中使用過的分佈索引鍵，則當 Amazon Redshift 判斷有更好的索引鍵時，則索引鍵可能會變更。

若要檢視資料表的排序索引鍵，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱 SVV_TABLE_INFO。若要檢視資料表的 Amazon Redshift Advisor 建議，請查詢 SVV_ALTER_TABLE_RECOMMENDATIONS 系統目錄檢視。如需詳細資訊，請參閱 SVV_ALTER_TABLE_RECOMMENDATIONS。若要檢視 Amazon Redshift 所採取的動作，請查詢 SVL_AUTO_WORKER_ACTION 系統目錄檢視。如需詳細資訊，請參閱 SVL_AUTO_WORKER_ACTION。

COMPOUND

指定使用複合索引鍵排序資料，該索引鍵是由列出的所有資料欄組成，並依其列出順序排列。當查詢依據排序資料欄的順序掃描資料列時，複合排序索引鍵最實用。當查詢依賴次要排序

資料欄時，使用複合索引鍵排序的效能優勢就會降低。您最多可為每個資料表定義 400 個 COMPOUND SORTKEY 資料欄。

INTERLEAVED

指定使用交錯排序索引鍵排序資料。最多可以為交錯排序索引鍵指定八個資料欄。

交錯排序索引鍵對排序索引鍵中的每個資料欄 (或資料欄子集) 都提供相等的權重，所以查詢不會取決於排序索引鍵中資料欄的順序。當查詢使用一個或多個次要排序資料欄時，交錯排序可大幅改善查詢效能。交錯排序在執行資料載入和清空操作時，會產生很少的額外負荷成本，

> **Important**
>
> 不要在具有依序增加屬性 (如身分資料欄、日期或時間戳記) 的資料欄上使用交錯排序索引鍵。

ENCODE AUTO

讓 Amazon Redshift 能夠自動調整資料表中所有資料欄的編碼類型，以最佳化查詢效能。ENCODE AUTO 會保留您在建立資料表時指定的初始編碼類型。然後，如果 Amazon Redshift 判斷新的編碼類型可以改善查詢效能，Amazon Redshift 就可以變更資料表資料欄的編碼類型。如果您沒有在資料表中的任何資料欄上指定編碼類型，ENCODE AUTO 就是預設值。

UNIQUE (column_name [,...])

此限制條件會指定，資料表中一個包含一個或多個資料欄的群組只能包含唯一值。唯一資料表限制條件的行為與資料欄限制條件的行為相同，但多了可橫跨多個資料欄的額外功能。在唯一限制條件的細節中，不會將 null 值視為相等。每個唯一資料表限制條件必須命名一組資料欄，這組資料欄有別於為資料表所定義的任何其他唯一或主索引鍵限制條件所命名的資料欄組。

> **Important**
>
> 唯一限制條件僅供參考，系統不會強制執行它們。

PRIMARY KEY (column_name [,...])

此限制條件會指定，資料表中的一個或多個資料欄只能包含唯一的 (不重複) 非 null 值。將一組資料欄識別為主索引鍵也會提供有關結構描述設計的中繼資料。主索引鍵表示，其他資料表可倚賴這組

資料欄做為資料列的唯一識別碼。一個資料表中可指定一個主索引鍵，無論是單一資料欄限制條件或資料表限制條件都可行。主索引鍵限制條件應命名一組資料欄，這組資料欄有別於由為相同資料表所定義的任何唯一限制條件命名的其他資料欄組。

> **Important**
>
> 主索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

FOREIGN KEY (column_name [, ...]) REFERENCES reftable [(refcolumn)]

此限制條件會指定外部索引鍵限制條件，其要求新資料表中一個包含一個或多個資料欄的群組包含的值，只能是符合參考資料表中某一資料列之參考資料欄的值。如果省略 refcolumn，則會使用 reftable 的主索引鍵。參考資料欄必須為參考資料表中唯一或主索引鍵限制條件的資料欄。

> **Important**
>
> 外部索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

使用須知

唯一性、主索引鍵和外部索引鍵限制僅供參考，Amazon Redshift 不會在您填入資料表時強制執行它們。例如，如果您將資料插入具有相依性的資料表中，即使插入違反限制也可以成功執行。儘管如此，主索引鍵和外部索引鍵仍會做為規劃提示，而且如果您的 ETL 程序或應用程式中的某些其他程序強制其完整性，則應該宣告它們。如需如何捨棄具有相依性之資料表的資訊，請參閱 DROP TABLE。

限制和配額

建立資料表時，請考量下列限制。

- 根據節點類型，叢集中的資料表數目上限有所限制。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的限制。
- 資料表名稱的字元數上限為 127 個。
- 單一資料表中可定義的資料欄數目上限為 1,600 個。
- 單一資料表中可定義的 SORTKEY 資料欄數目上限為 400 個。

欄位層級設定和資料表層級設定的摘要

在資料欄層級或資料表層級可設定數種屬性和設定。某些情況下，在資料欄層級或資料表層級設定屬性或限制條件的效果相同。而有些情況下則會產生不同的結果。

下列清單摘要說明資料欄層級和資料表層級設定：

DISTKEY

在資料欄層級或資料表層級設定的效果並無差異。

如果設定了 DISTKEY，無論是在資料欄層級或資料表層級，DISTSTYLE 都必須設定為 KEY，或完全不設定。DISTSTYLE 只能在資料表層級設定。

SORTKEY

若在資料欄層級設定，SORTKEY 必須是單一資料欄。若在資料表層級設定 SORTKEY，可由一個或多個資料欄組成複合或交錯的複合排序索引鍵。

COLLATE CASE_SENSITIVE | COLLATE CASE_INSENSITIVE

Amazon Redshift 不支援修改資料欄的區分大小寫組態。當您將新的資料欄附加到資料表時，Amazon Redshift 會使用預設值來區分大小寫。Amazon Redshift 在附加新的資料欄時不支援 COLLATE 關鍵字。

如需如何使用資料庫定序建立資料庫的資訊，請參閱 CREATE DATABASE。

如需 COLLATE 函數的詳細資訊，請參閱 COLLATE 函數。

UNIQUE

在資料欄層級可將一個或多個索引鍵設定為 UNIQUE；UNIQUE 限制條件會分別套用至每個資料欄。若在資料表層級設定 UNIQUE，可由一個或多個資料欄組成複合 UNIQUE 限制條件。

PRIMARY KEY

若在資料欄層級設定，PRIMARY KEY 必須是單一資料欄。若在資料表層級設定 PRIMARY KEY，可由一個或多個資料欄組成複合主索引鍵。

FOREIGN KEY

無論 FOREIGN KEY 是在資料欄層級或資料表層級設定，其效果並無差異。在資料欄層級的語法為單純的 REFERENCES reftable [(refcolumn)]。

傳入資料的分佈

當傳入資料的雜湊分佈機制與目標資料表的機制相同時，不需要在資料載入時實際分佈資料。例如，如果新資料表設定了分佈索引鍵，而且要從另一個分佈在相同索引鍵資料欄上的資料表插入資料，則會使用相同的節點和分割就地載入資料。不過，如果來源和目標資料表都設定為 EVEN 分佈，則資料會重新分佈至目標資料表中。

寬資料表

您或許能夠建立非常寬的資料表，但無法在資料表上執行查詢處理，像是 INSERT 或 SELECT 陳述式。有固定寬度資料欄 (如 CHAR) 的資料表寬度上限為 64KB - 1 (或 65535 個位元組)。如果資料表包含 VARCHAR 資料欄，資料表就能擁有較大的宣告寬度，而不會傳回錯誤，因為 VARCHARS 資料欄的宣告寬度不會完全計入計算出的查詢處理限制。VARCHAR 資料欄的有效查詢處理限制會因為一些因素而不同。

如果資料表對於插入或選取操作來說太寬，您會收到下列錯誤。

```
ERROR:  8001
DETAIL:  The combined length of columns processed in the SQL statement
exceeded the query-processing limit of 65535 characters (pid:7627)
```

範例

如需示範如何使用 CREATE TABLE 命令的範例，請參閱 範例 主題。

範例

以下範例示範 Amazon Redshift CREATE TABLE 陳述式中各種不同的資料欄和資料表屬性。如需 CREATE TABLE 的相關資訊，包括參數定義，請參閱 CREATE TABLE。

許多範例都使用 TICKIT 範例資料集中的資料表和資料。如需詳細資訊，請參閱 tz 資料庫。

您可以在 CREATE TABLE 命令中使用資料庫名稱和結構描述名稱作為資料表名稱的字首。例如：`dev_database.public.sales`。資料庫名稱必須是您所連線的資料庫。任何在另一個資料庫中建立資料庫物件的嘗試都會失敗，並顯示操作無效錯誤。

建立含有分佈索引鍵、複合排序索引鍵和壓縮的資料表

下列範例會在 TICKIT 資料庫中建立 SALES 資料表，並且為數個資料欄定義壓縮。LISTID 會宣告為分佈索引鍵，而 LISTID 和 SELLERID 會宣告為多資料欄複合排序索引鍵。另外也會為資料表定義主索引鍵和外部索引鍵限制條件。在範例中建立資料表之前，如果限制條件不存在，您可能需要將 UNIQUE 條件限制新增至外部索引鍵參考的每個資料行。

```
create table sales(
salesid integer not null,
listid integer not null,
sellerid integer not null,
buyerid integer not null,
eventid integer not null encode mostly16,
dateid smallint not null,
qtysold smallint not null encode mostly8,
pricepaid decimal(8,2) encode delta32k,
commission decimal(8,2) encode delta32k,
saletime timestamp,
primary key(salesid),
foreign key(listid) references listing(listid),
foreign key(sellerid) references users(userid),
foreign key(buyerid) references users(userid),
foreign key(dateid) references date(dateid))
distkey(listid)
compound sortkey(listid,sellerid);
```

結果如下：

```
schemaname | tablename | column     | type                          | encoding | distkey
 | sortkey | notnull
-----------+-----------+------------+-------------------------------+----------+---------
+---------+--------
public     | sales     | salesid    | integer                       | lzo      | false
 |       0 | true
public     | sales     | listid     | integer                       | none     | true
 |       1 | true
public     | sales     | sellerid   | integer                       | none     | false
 |       2 | true
public     | sales     | buyerid    | integer                       | lzo      | false
 |       0 | true
public     | sales     | eventid    | integer                       | mostly16 | false
 |       0 | true
public     | sales     | dateid     | smallint                      | lzo      | false
 |       0 | true
public     | sales     | qtysold    | smallint                      | mostly8  | false
 |       0 | true
public     | sales     | pricepaid  | numeric(8,2)                  | delta32k | false
 |       0 | false
public     | sales     | commission | numeric(8,2)                  | delta32k | false
 |       0 | false
```

```
public     | sales     | saletime   | timestamp without time zone | lzo      | false
 |       0 | false
```

下列範例會使用不區分大小寫的資料欄 col1 建立資料表 t1。

```
create table T1 (
  col1 Varchar(20) collate case_insensitive
 );

insert into T1 values ('bob'), ('john'), ('Tom'), ('JOHN'), ('Bob');
```

查詢資料表：

```
select * from T1 where col1 = 'John';

col1
------
 john
 JOHN
(2 rows)
```

使用交錯排序索引鍵建立資料表

以下範例會建立具有交錯排序索引鍵的 CUSTOMER 資料表。

```
create table customer_interleaved (
  c_custkey      integer        not null,
  c_name         varchar(25)    not null,
  c_address      varchar(25)    not null,
  c_city         varchar(10)    not null,
  c_nation       varchar(15)    not null,
  c_region       varchar(12)    not null,
  c_phone        varchar(15)    not null,
  c_mktsegment      varchar(10)    not null)
diststyle all
interleaved sortkey (c_custkey, c_city, c_mktsegment);
```

使用 IF NOT EXISTS 建立資料表

下列範例會建立 CITIES 資料表，或不執行任何動作，並於該資料表已存在時傳回訊息：

```
create table if not exists cities(
```

```
cityid integer not null,
city varchar(100) not null,
state char(2) not null);
```

建立採用 ALL 分佈的資料表

以下範例會建立採用 ALL 分佈的 VENUE 資料表。

```
create table venue(
venueid smallint not null,
venuename varchar(100),
venuecity varchar(30),
venuestate char(2),
venueseats integer,
primary key(venueid))
diststyle all;
```

建立採用 EVEN 分佈的資料表

以下範例會建立名為 MYEVENT 且包含三個資料欄的資料表。

```
create table myevent(
eventid int,
eventname varchar(200),
eventcity varchar(30))
diststyle even;
```

資料表會均勻分佈且不會排序。資料表沒有宣告的 DISTKEY 或 SORTKEY 欄。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'myevent';

  column   |          type          | encoding | distkey | sortkey
-----------+------------------------+----------+---------+---------
 eventid   | integer                | lzo      | f       |       0
 eventname | character varying(200) | lzo      | f       |       0
 eventcity | character varying(30)  | lzo      | f       |       0
(3 rows)
```

建立 LIKE (類似) 另一個資料表的暫時資料表

以下範例會建立名為 TEMPEVENT 的臨時資料表，它的資料欄繼承自 EVENT 資料表。

```
create temp table tempevent(like event);
```

此資料表也會繼承其父資料表的 DISTKEY 和 SORTKEY 屬性：

```
select "column", type, encoding, distkey, sortkey
 from pg_table_def where tablename = 'tempevent';

  column   |            type             | encoding | distkey | sortkey
-----------+-----------------------------+----------+---------+---------
 eventid   | integer                     | none     | t       |       1
 venueid   | smallint                    | none     | f       |       0
 catid     | smallint                    | none     | f       |       0
 dateid    | smallint                    | none     | f       |       0
 eventname | character varying(200)      | lzo      | f       |       0
 starttime | timestamp without time zone | bytedict | f       |       0
(6 rows)
```

建立具有 IDENTITY 欄位的資料表

以下範例會建立名為 VENUE_IDENT 的資料表，其中有一個名為 VENUEID 的 IDENTITY 資料欄。此資料欄從 0 開始，並隨每筆記錄遞增 1。VENUEID 也會宣告為資料表的主索引鍵。

```
create table venue_ident(venueid bigint identity(0, 1),
venuename varchar(100),
venuecity varchar(30),
venuestate char(2),
venueseats integer,
primary key(venueid));
```

建立具有預設 IDENTITY 欄位的資料表

以下範例會建立名為 t1 的資料表。此資料表具有名為 hist_id 的 IDENTITY 資料欄，以及名為 base_id 的預設 IDENTITY 資料欄。

```
CREATE TABLE t1(
  hist_id BIGINT IDENTITY NOT NULL, /* Cannot be overridden */
  base_id BIGINT GENERATED BY DEFAULT AS IDENTITY NOT NULL, /* Can be overridden */
  business_key varchar(10) ,
  some_field varchar(10)
);
```

將資料列插入至資料表顯示已同時產生 hist_id 及 base_id 值。

```
INSERT INTO T1 (business_key, some_field) values ('A','MM');
```

```
SELECT * FROM t1;

 hist_id | base_id | business_key | some_field
---------+---------+--------------+------------
       1 |       1 | A            | MM
```

插入第二列顯示已產生 base_id 的預設值。

```
INSERT INTO T1 (base_id, business_key, some_field) values (DEFAULT, 'B','MNOP');
```

```
SELECT * FROM t1;

 hist_id | base_id | business_key | some_field
---------+---------+--------------+------------
       1 |       1 | A            | MM
       2 |       2 | B            | MNOP
```

插入第三列顯示 base_id 的值不需要是唯一的。

```
INSERT INTO T1 (base_id, business_key, some_field) values (2,'B','MNNN');
```

```
SELECT * FROM t1;

 hist_id | base_id | business_key | some_field
---------+---------+--------------+------------
       1 |       1 | A            | MM
       2 |       2 | B            | MNOP
       3 |       2 | B            | MNNN
```

建立具有 DEFAULT 欄位的資料表

下列範例會建立 CATEGORYDEF 資料表，它會宣告每個資料欄的預設值：

```
create table categorydef(
catid smallint not null default 0,
```

```
catgroup varchar(10) default 'Special',
catname varchar(10) default 'Other',
catdesc varchar(50) default 'Special events',
primary key(catid));

insert into categorydef values(default,default,default,default);
```

```
select * from categorydef;

 catid | catgroup | catname |    catdesc
-------+----------+---------+----------------
     0 | Special  | Other   | Special events
(1 row)
```

DISTSTYLE、DISTKEY 和 SORTKEY 選項

下列範例說明 DISTKEY、SORTKEY 和 DISTSTYLE 選項的運作方式。在此範例中，COL1 是分佈索引鍵，因此分佈樣式必須設定為 KEY，或是不設定。根據預設，資料表沒有排序索引鍵，因此不會排序：

```
create table t1(col1 int distkey, col2 int) diststyle key;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't1';

column |  type   | encoding | distkey | sortkey
-------+---------+----------+---------+---------
col1   | integer | az64     | t       | 0
col2   | integer | az64     | f       | 0
```

在下列範例中，會將同一個資料欄定義為分佈索引鍵和排序索引鍵。同樣地，分佈樣式必須設定為 KEY，或是不設定。

```
create table t2(col1 int distkey sortkey, col2 int);
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't2';

column |  type   | encoding | distkey | sortkey
-------+---------+----------+---------+---------
```

```
col1   | integer | none     | t       | 1
col2   | integer | az64     | f       | 0
```

在下列範例中，不會將任何資料欄設定為分佈索引鍵，COL2 會設定為排序索引鍵，而分佈樣式會設定為 ALL：

```
create table t3(col1 int, col2 int sortkey) diststyle all;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't3';

Column |  Type   | Encoding | DistKey | SortKey
-------+---------+----------+---------+--------
col1   | integer | az64     | f       | 0
col2   | integer | none     | f       | 1
```

在下列範例中，分佈樣式會設定為 EVEN，但不會明確定義任何排序索引鍵；因此，資料表會均勻分佈，但不會排序。

```
create table t4(col1 int, col2 int) diststyle even;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't4';

             column |  type   |encoding | distkey | sortkey
--------+---------+---------+---------+--------
col1    | integer | az64    | f       | 0
col2    | integer | az64    | f       | 0
```

使用 ENCODE AUTO 選項建立資料表

下列範例會建立具有自動壓縮編碼的 t1 資料表。如果您沒有在任何資料欄上指定編碼類型，ENCODE AUTO 就是資料表的預設值。

```
create table t1(c0 int, c1 varchar);
```

下列範例會透過指定 ENCODE AUTO 建立具有自動壓縮編碼的 t2 資料表。

```
create table t2(c0 int, c1 varchar) encode auto;
```

下列範例會透過指定 ENCODE AUTO 建立具有自動壓縮編碼的 t3 資料表。資料欄 c0 是以初始編碼類型 DELTA 所定義。如果其他編碼提供更好的查詢效能，Amazon Redshift 可以變更編碼。

```
create table t3(c0 int encode delta, c1 varchar) encode auto;
```

下列範例會透過指定 ENCODE AUTO 建立具有自動壓縮編碼的 t4 資料表。資料欄 c0 是以初始編碼 DELTA 來定義，而資料欄 c1 則以 LZO 初始編碼來定義。如果其他編碼提供更好的查詢效能，Amazon Redshift 可以變更這些編碼。

```
create table t4(c0 int encode delta, c1 varchar encode lzo) encode auto;
```

CREATE TABLE AS

主題

根據查詢建立新資料表。此資料表的擁有者是發出命令的使用者。

載入的新資料表包含命令中的查詢所定義的資料。資料表資料欄的名稱和資料類型會與查詢的輸出資料欄相關聯。CREATE TABLE AS (CTAS) 命令會建立新資料表，並評估查詢以載入新資料表。

語法

```
CREATE [ [ LOCAL ] { TEMPORARY | TEMP } ]
TABLE table_name
[ ( column_name [, ... ] ) ]
[ BACKUP { YES | NO } ]
[ table_attributes ]
AS query

where table_attributes are:
[ DISTSTYLE { AUTO | EVEN | ALL | KEY } ]
[ DISTKEY( distkey_identifier ) ]
[ [ COMPOUND | INTERLEAVED ] SORTKEY( column_name [, ...] ) ]
```

參數

LOCAL

雖然陳述式中可接受此選用關鍵字，但是在 Amazon Redshift 中沒有作用。

TEMPORARY | TEMP

建立臨時資料表。臨時資料表會在其建立所在的工作階段結束時自動捨棄。

table_name

要建立的資料表名稱。

> **Important**
>
> 若您指定 '#' 開頭的資料表名稱，所建立的資料表會是臨時資料表。例如：
>
> ```
> create table #newtable (id) as select * from oldtable;
> ```

資料表名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。Amazon Redshift 會依節點類型強制執行每個叢集的資料表數量配額限制。資料表名稱也可透過資料庫和結構描述名稱來限定，如下表所示。

```
create table tickit.public.test (c1) as select * from oldtable;
```

在此範例中，`tickit` 是資料庫名稱，而 `public` 是結構描述名稱。如果資料庫或結構描述不存在，則陳述式會傳回錯誤。

若提供結構描述名稱，則會在該結構描述中建立新資料表 (假設建立者具有存取結構描述的權限)。資料表名稱對於該結構描述來說必須是唯一的。如果未指定結構描述，則會使用目前資料庫結構描述建立資料表。如果您要建立臨時資料表，就不能指定結構描述名稱，因為臨時資料表會採用特殊結構描述。

若臨時資料表是在不同的工作階段中建立的話，則允許在同一個資料庫中同時有多個同名的臨時資料表存在。這些資料表會指派至不同的結構描述。

column_name

新資料表中資料欄的名稱。若未提供任何資料欄名稱，則會採用查詢的輸出資料欄名稱做為資料欄名稱。表達式會使用預設資料欄名稱。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

BACKUP { YES | NO }

此子句會指定資料表是否應包含在自動化和手動叢集快照中。若是像臨時資料表這類不會包含重要資料的資料表，指定 BACKUP NO 可以在建立快照以及從快照還原時節省處理時間，並減少 Amazon Simple Storage Service 上的儲存空間。BACKUP NO 設定對於將資料自動複寫到叢集內其他節點的操作並無影響，因此指定了 BACKUP NO 的資料表會在節點發生故障時還原。預設值為 BACKUP YES。

DISTSTYLE { AUTO | EVEN | KEY | ALL }

定義整個資料表的資料分佈樣式。Amazon Redshift 會根據資料表上指定的分佈樣式，將資料表的資料列分散到運算節點。預設值為 DISTSTYLE AUTO。

您為資料表選取的分佈樣式會影響資料庫的整體效能。如需詳細資訊，請參閱 使用資料分佈樣式。

- AUTO：Amazon Redshift 會根據資料表資料來指派最佳分佈樣式。若要檢視套用至資料表的分佈樣式，請查詢 PG_CLASS 系統目錄資料表。如需詳細資訊，請參閱 檢視分佈樣式。
- EVEN：資料表中的資料會採循環分佈的方式，均勻分配到叢集中的各個節點。資料列 ID 會用來決定分佈，並且將大致相同的資料列數分佈到每個節點。這是預設分佈方法。
- KEY：資料是依 DISTKEY 資料欄中的值分佈。當您將聯結資料表的聯結資料欄設定為分佈索引鍵時，兩個資料表的聯結資料列會在運算節點上並存。資料並存時，最佳化工具就能更有效率地執行聯結。若您指定 DISTSTYLE KEY，則必須命名一個 DISTKEY 資料欄。
- ALL：整個資料表的副本分佈至每個節點。此分佈樣式可確保每個節點上都有任何聯結所需的所有資料列，但儲存空間的需求也會倍增，並且會增加資料表的負載和維護次數。ALL 分佈在 KEY 分佈不適用的情況下搭配特定維度資料表使用時，可改善執行時間，但必須在效能提升與維護成本之間進行權衡。

DISTKEY (column)

指定分佈索引鍵的資料欄名稱或位置編號。使用資料表的選用資料欄清單中所指定的名稱，或是查詢的選取清單中指定的名稱。或者，使用位置編號，其中選取的第一個資料欄為 1、第二個為 2，以此類推。資料表中只能有一個資料欄是分佈索引鍵：

- 如果您將資料欄宣告為 DISTKEY 資料欄，則 DISTSTYLE 必須設定為 KEY，或完全不設定。
- 如果您未宣告 DISTKEY 資料欄，則可以將 DISTSTYLE 設定為 EVEN。
- 如果您不指定 DISTKEY 或 DISTSTYLE，則 CTAS 會根據 SELECT 子句的查詢計畫判斷新資料表的分佈樣式。如需詳細資訊，請參閱 欄位和資料表屬性的繼承。

您可以將同一個資料欄定義為分佈索引鍵和排序索引鍵；此方法主要目的是在所指的資料欄要與查詢中的資料欄聯結時加速聯結。

[COMPOUND | INTERLEAVED] SORTKEY (column_name [, ...])

指定資料表的一個或多個排序索引鍵。當資料載入資料表時，資料會依指定為排序索引鍵的資料欄排序。

您也可以選擇指定 COMPOUND 或 INTERLEAVED 排序樣式。預設值為 COMPOUND。如需詳細資訊，請參閱 使用排序索引鍵。

您最多可為每個資料表定義 400 個 COMPOUND SORTKEY 資料欄或 8 個 INTERLEAVED SORTKEY 資料欄。

如果您不指定 SORTKEY，則 CTAS 會根據 SELECT 子句的查詢計畫判斷新資料表的排序索引鍵。如需詳細資訊，請參閱 欄位和資料表屬性的繼承。

COMPOUND

指定使用複合索引鍵排序資料，該索引鍵是由列出的所有資料欄組成，並依其列出順序排列。當查詢依據排序資料欄的順序掃描資料列時，複合排序索引鍵最實用。當查詢依賴次要排序資料欄時，使用複合索引鍵排序的效能優勢就會降低。您最多可為每個資料表定義 400 個 COMPOUND SORTKEY 資料欄。

INTERLEAVED

指定使用交錯排序索引鍵排序資料。最多可以為交錯排序索引鍵指定八個資料欄。

交錯排序索引鍵對排序索引鍵中的每個資料欄 (或資料欄子集) 都提供相等的權重，所以查詢不會取決於排序索引鍵中資料欄的順序。當查詢使用一個或多個次要排序資料欄時，交錯排序可大幅改善查詢效能。交錯排序在執行資料載入和清空操作時，會產生很少的額外負荷成本，

AS query

Amazon Redshift 支援的任何查詢 (SELECT 陳述式)。

CTAS 使用須知

限制

Amazon Redshift 會依節點類型強制執行每個叢集的資料表數量配額限制。

資料表名稱的字元數上限為 127 個。

單一資料表中可定義的資料欄數目上限為 1,600 個。

欄位和資料表屬性的繼承

CREATE TABLE AS (CTAS) 資料表不會從其建立的來源資料表繼承限制條件、身分資料欄、預設資料欄值或主索引鍵。

您無法指定 CTAS 資料表的資料欄壓縮編碼。Amazon Redshift 會自動指派壓縮編碼，如下所示：

- 定義為排序索引鍵的資料欄會有指派的 RAW 壓縮。
- 定義為 BOOLEAN、REAL、DOUBLE PRECISION、GEOMETRY 或 GEOGRAPHY 資料類型的資料行會有指派的 RAW 壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP 或 TIMESTAMPTZ 的資料欄會有指派的 AZ64 壓縮。
- 定義為 CHAR、VARCHAR 或 VARBYTE 的資料欄會有指派的 LZO 壓縮。

如需詳細資訊，請參閱 壓縮編碼 及 資料類型。

若要明確指派資料欄編碼，請使用 CREATE TABLE。

CTAS 會根據 SELECT 子句的查詢計畫判斷新資料表的分佈樣式和排序索引鍵。

若是複雜的查詢，像是包含聯結、彙總、order by 子句或限制子句的查詢，CTAS 會根據查詢計畫盡可能選擇最佳分佈樣式和排序索引鍵。

> **Note**
>
> 為了在大型資料集或複雜查詢上發揮最佳效能，建議您使用一般資料集進行測試。

您可藉由查看查詢計畫來了解查詢最佳化工具選擇哪些資料欄 (如有的話) 來排序和分佈資料，如此經常能夠預測出 CTAS 會選擇哪個分佈索引鍵和排序索引鍵。如果查詢計畫的頂端節點是來自單一資料表的簡單循序掃描 (XN Seq Scan)，則 CTAS 通常會使用來源資料表的分佈樣式和排序索引鍵。如果查詢計劃的頂端節點是任何其他連續掃描 (例如 XN Limit、XN 排序 HashAggregate、XN 等)，CTAS 會根據查詢計劃選擇最佳的分佈樣式和排序索引鍵。

例如，假設您使用下列類型的 SELECT 子句建立五個資料表：

- 簡單 select 陳述式
- 限制子句

- 使用 LISTID 的 order by 子句
- 使用 QTYSOLD 的 order by 子句
- 含有 group by 子句的 SUM 彙總函數。

下列範例顯示每個 CTAS 陳述式的查詢計畫。

```
explain create table sales1_simple as select listid, dateid, qtysold from sales;
                          QUERY PLAN
-------------------------------------------------------------
 XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(1 row)


explain create table sales2_limit as select listid, dateid, qtysold from sales limit
 100;
                             QUERY PLAN
---------------------------------------------------------------------
 XN Limit  (cost=0.00..1.00 rows=100 width=8)
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(2 rows)


explain create table sales3_orderbylistid as select listid, dateid, qtysold from sales
 order by listid;
                              QUERY PLAN
----------------------------------------------------------------------------
 XN Sort  (cost=1000000016724.67..1000000017155.81 rows=172456 width=8)
   Sort Key: listid
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(3 rows)


explain create table sales4_orderbyqty as select listid, dateid, qtysold from sales
 order by qtysold;
                              QUERY PLAN
----------------------------------------------------------------------------
 XN Sort  (cost=1000000016724.67..1000000017155.81 rows=172456 width=8)
   Sort Key: qtysold
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(3 rows)
```

```
explain create table sales5_groupby as select listid, dateid, sum(qtysold) from sales
 group by listid, dateid;
                            QUERY PLAN
------------------------------------------------------------------------
 XN HashAggregate  (cost=3017.98..3226.75 rows=83509 width=8)
   ->  XN Seq Scan on sales  (cost=0.00..1724.56 rows=172456 width=8)
(2 rows)
```

若要檢視每個資料表的分佈索引鍵和排序索引鍵，請查詢 PG_TABLE_DEF 系統目錄資料表，如下所示。

```
select * from pg_table_def where tablename like 'sales%';

      tablename       |   column   | distkey | sortkey
----------------------+------------+---------+---------
 sales                | salesid    | f       |       0
 sales                | listid     | t       |       0
 sales                | sellerid   | f       |       0
 sales                | buyerid    | f       |       0
 sales                | eventid    | f       |       0
 sales                | dateid     | f       |       1
 sales                | qtysold    | f       |       0
 sales                | pricepaid  | f       |       0
 sales                | commission | f       |       0
 sales                | saletime   | f       |       0
 sales1_simple        | listid     | t       |       0
 sales1_simple        | dateid     | f       |       1
 sales1_simple        | qtysold    | f       |       0
 sales2_limit         | listid     | f       |       0
 sales2_limit         | dateid     | f       |       0
 sales2_limit         | qtysold    | f       |       0
 sales3_orderbylistid | listid     | t       |       1
 sales3_orderbylistid | dateid     | f       |       0
 sales3_orderbylistid | qtysold    | f       |       0
 sales4_orderbyqty    | listid     | t       |       0
 sales4_orderbyqty    | dateid     | f       |       0
 sales4_orderbyqty    | qtysold    | f       |       1
 sales5_groupby       | listid     | f       |       0
 sales5_groupby       | dateid     | f       |       0
 sales5_groupby       | sum        | f       |       0
```

下表顯示結果摘要。為了簡化起見，我們省略了說明計畫中的成本、資料列和寬度詳細資訊。

資料表	CTAS SELECT 陳述式	說明計劃頂端節點	分佈索引鍵	排序索引鍵
S1_SIMPLE	select listid, dateid, qtysold from sales	XN Seq Scan on sales ...	LISTID	DATEID
S2_LIMIT	select listid, dateid, qtysold from sales limit 100	XN Limit ...	無 (EVEN)	無
S3_ORDER_BY_LISTID	select listid, dateid, qtysold from sales order by listid	XN Sort ... Sort Key: listid	LISTID	LISTID
S4_ORDER_BY_QTY	select listid, dateid, qtysold from sales order by qtysold	XN Sort ... Sort Key: qtysold	LISTID	QTYSOLD
S5_GROUP_BY	select listid, dateid, sum(qtysold) from sales group by listid, dateid	XN HashAggregate ...	無 (EVEN)	無

您可以在 CTAS 陳述式中明確指定分佈樣式和排序索引鍵。例如，以下陳述式會使用 EVEN 分佈建立資料表，並指定 SALESID 做為排序索引鍵。

```
create table sales_disteven
diststyle even
sortkey (salesid)
as
select eventid, venueid, dateid, eventname
from event;
```

壓縮編碼

ENCODE AUTO 會用來作為資料表的預設值。Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。

傳入資料的分佈

當傳入資料的雜湊分佈機制與目標資料表的機制相同時，不需要在資料載入時實際分佈資料。例如，如果新資料表設定了分佈索引鍵，而且要從另一個分佈在相同索引鍵資料欄上的資料表插入資料，則會使用相同的節點和分割就地載入資料。不過，如果來源和目標資料表都設定為 EVEN 分佈，則資料會重新分佈至目標資料表中。

自動 ANALYZE 操作

Amazon Redshift 會自動分析您使用 CTAS 命令建立的資料表。您不需要在初次建立這些資料表時，對其執行 ANALYZE 命令。若您修改這些資料表，則應依照與其他資料表相同的方式進行分析。

CTAS 範例

以下範例會為 EVENT 資料表建立名為 EVENT_BACKUP 的資料表：

```
create table event_backup as select * from event;
```

產生的資料表會從 EVENT 資料表繼承分佈和排序索引鍵。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'event_backup';

column    | type                        | encoding | distkey | sortkey
----------+-----------------------------+----------+---------+--------
catid     | smallint                    | none     | false   |       0
dateid    | smallint                    | none     | false   |       1
eventid   | integer                     | none     | true    |       0
eventname | character varying(200)      | none     | false   |       0
starttime | timestamp without time zone | none     | false   |       0
venueid   | smallint                    | none     | false   |       0
```

以下命令會從 EVENT 資料表選取四個資料欄，藉此建立名為 EVENTDISTSORT 的新資料表。新資料表會依 EVENTID 分佈並依 EVENTID 和 DATEID 排序：

```
create table eventdistsort
distkey (1)
```

```
sortkey (1,3)
as
select eventid, venueid, dateid, eventname
from event;
```

結果如下所示：

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdistsort';

column   |          type          | encoding | distkey | sortkey
---------+------------------------+----------+---------+-------
eventid   | integer                | none     | t       | 1
venueid   | smallint               | none     | f       | 0
dateid    | smallint               | none     | f       | 2
eventname | character varying(200)| none     | f       | 0
```

您可以使用分佈和排序索引鍵的資料欄名稱建立完全相同的資料表。例如：

```
create table eventdistsort1
distkey (eventid)
sortkey (eventid, dateid)
as
select eventid, venueid, dateid, eventname
from event;
```

以下陳述式會將均勻分佈套用至資料表，但不會定義明確的排序索引鍵。

```
create table eventdisteven
diststyle even
as
select eventid, venueid, dateid, eventname
from event;
```

資料表不會從 EVENT 資料表 (EVENTID) 繼承排序索引鍵，因為針對新資料表指定了 EVEN 分佈。新資料表沒有排序索引鍵，也沒有分佈索引鍵。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdisteven';

column    |          type          | encoding | distkey | sortkey
```

```
----------+------------------------+----------+---------+---------
eventid   | integer                | none     | f       | 0
venueid   | smallint               | none     | f       | 0
dateid    | smallint               | none     | f       | 0
eventname | character varying(200) | none     | f       | 0
```

以下陳述式會套用均勻分佈，並定義排序索引鍵：

```
create table eventdistevensort diststyle even sortkey (venueid)
as select eventid, venueid, dateid, eventname from event;
```

產生的資料表有排序索引鍵，但沒有分佈索引鍵。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdistevensort';

column    |          type          | encoding | distkey | sortkey
----------+------------------------+----------+---------+-------
eventid   | integer                | none     | f       | 0
venueid   | smallint               | none     | f       | 1
dateid    | smallint               | none     | f       | 0
eventname | character varying(200) | none     | f       | 0
```

以下陳述式會依傳入資料 (依 EVENTID 排序) 的不同索引鍵資料欄重新分佈 EVENT 資料表，並且不會定義 SORTKEY 資料欄；因此資料表不會排序。

```
create table venuedistevent distkey(venueid)
as select * from event;
```

結果如下所示：

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'venuedistevent';

 column   |            type             | encoding | distkey | sortkey
----------+-----------------------------+----------+---------+-------
eventid   | integer                     | none     | f       | 0
venueid   | smallint                    | none     | t       | 0
catid     | smallint                    | none     | f       | 0
dateid    | smallint                    | none     | f       | 0
eventname | character varying(200)      | none     | f       | 0
```

```
starttime | timestamp without time zone | none     | f       | 0
```

CREATE USER

建立新的資料庫使用者。視權限和角色而定，資料庫使用者可以擷取資料、執行命令，以及在資料庫中執行其他動作。您必須是資料庫超級使用者才能執行此命令。

所需權限

以下是 CREATE USER 所需的權限：

- 超級使用者
- 具有 CREATE USER 權限的使用者

語法

```
CREATE USER name [ WITH ]
PASSWORD { 'password' | 'md5hash' | 'sha256hash' | DISABLE }
[ option [ ... ] ]

where option can be:

CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }
| IN GROUP groupname [, ... ]
| VALID UNTIL 'abstime'
| CONNECTION LIMIT { limit | UNLIMITED }
| SESSION TIMEOUT limit
| EXTERNALID external_id
```

參數

name

欲建立的使用者名稱。使用者名稱不可以是 PUBLIC。如需有效名稱的相關資訊，請參閱 名稱與識別碼。

WITH

選用的關鍵字。WITH 會被 Amazon Redshift 忽略

PASSWORD { 'password' | 'md5hash' | 'sha256hash' | DISABLE }

設定使用者的密碼。

根據預設，使用者可以變更自己擁有的密碼，除非密碼遭到停用。若要停用使用者的密碼，請指定 DISABLE。停用使用者密碼時，密碼會從系統中刪除，使用者只能使用臨時 AWS Identity and Access Management (IAM) 使用者登入資料登入。如需詳細資訊，請參閱使用 IAM 身分驗證來產生資料庫使用者登入資料。只有超級使用者能夠啟用或停用密碼，您無法停用超級使用者的密碼。若要啟用密碼，請執行 ALTER USER 並指定密碼。

您可以以純文字、MD5 雜湊字串或 SHA256 雜湊字串形式指定密碼。

> **Note**
>
> 當您使用、或 Amazon Redshift API 啟動新叢集時 AWS Management Console AWS CLI，您必須為初始資料庫使用者提供純文字密碼。您之後可以使用 ALTER USER 變更密碼。

若是純文字，密碼必須滿足下列限制條件：

- 長度必須為 8 到 64 個字元。
- 至少須包含一個大寫字母、一個小寫字母和一個數字。
- 它可以使用任何 ASCII 字元 (ASCII 碼 33–126)，但 ' (單引號)、" (雙引號)、\、/ 或 @ 除外。

除了以純文字傳遞 CREATE USER 密碼參數之外，更安全的替代方法是指定包括密碼和使用者名稱的 MD5 雜湊字串。

> **Note**
>
> 當您指定 MD5 雜湊字串時，CREATE USER 命令會檢查有效的 MD5 雜湊字串，但不會驗證字串的密碼部分。在此情況下可能會建立密碼，例如空字串，但此密碼無法用來登入資料庫。

若要指定 MD5 密碼，請依照下列步驟執行：

1. 串連密碼和使用者名稱。

   例如，密碼 `ez` 和使用者 `user1` 的串連字串為 `ezuser1`。

2. 將串連字串轉換成 32 個字元的 MD5 雜湊字串。您可以使用任何 MD5 公用程式來建立雜湊字串。以下範例會使用 Amazon Redshift MD5 函數 和串連運算子 (||) 傳回 32 個字元的 MD5 雜湊字串。

```
select md5('ez' || 'user1');

md5
--------------------------------
153c434b4b77c89e6b94f12c5393af5b
```

3. 在 MD5 雜湊字串前方串連 'md5' 並提供串連字串做為 md5hash 引數。

```
create user user1 password 'md5153c434b4b77c89e6b94f12c5393af5b';
```

4. 使用登入憑證登入資料庫。

   例如，以 user1 的身分和密碼 ez 登入。

另一個安全的替代方法是指定密碼字串的 SHA-256 雜湊，或是您可以提供自己的有效 SHA-256 摘要和用於建立摘要的 256 位元 Salt。

- 摘要 — 雜湊函數的輸出。
- Salt — 隨機產生且可與密碼結合的資料，有助於減少雜湊函數輸出中的模式。

```
'sha256|Mypassword'
```

```
'sha256|digest|256-bit-salt'
```

在下列範例中，Amazon Redshift 會產生並管理 Salt。

```
CREATE USER admin PASSWORD 'sha256|Mypassword1';
```

下列範例會提供有效 SHA-256 摘要和用來建立摘要的 256 位元 Salt。

要指定密碼並使用自己的 Salt 對其進行雜湊，請按照以下步驟操作：

1. 建立 256 位元 Salt。您可以透過使用任何十六進位字串產生器產生 64 個字元長的字串來取得 Salt。在此範例中，Salt 是 c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6。

2. 使用 FROM_HEX 函數將您的 Salt 轉換為二進位。這是因為 SHA2 函數需要 Salt 的二進位表示法。請參閱以下陳述式。

```
SELECT
 FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6');
```

3. 使用 CONCAT 函數將您的 Salt 附加到您的密碼中。在此範例中，密碼為 Mypassword1。請參閱以下陳述式。

```
SELECT
 CONCAT('Mypassword1',FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f951303
```

4. 使用 SHA2 函數從您的密碼和 salt 組合中建立摘要。請參閱以下陳述式。

```
SELECT
 SHA2(CONCAT('Mypassword1',FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9
 0);
```

5. 使用先前步驟中的摘要和 Salt，建立使用者。請參閱以下陳述式。

```
CREATE USER admin PASSWORD 'sha256|
821708135fcc42eb3afda85286dee0ed15c2c461d000291609f77eb113073ec2|
c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6';
```

6. 使用登入憑證登入資料庫。

   例如，以 admin 的身分和密碼 Mypassword1 登入。

如果您在未指定雜湊函數的情況下，以純文字格式設定密碼，則會使用使用者名稱作為 Salt 來產生 MD5 摘要。

CREATEDB | NOCREATEDB

CREATEDB 選項可讓新使用者建立資料庫。預設值是 NOCREATEDB。

CREATEUSER | NOCREATEUSER

CREATEUSER 選項可建立具有所有資料庫權限的超級使用者，包括 CREATE USER。預設值為 NOCREATEUSER。如需詳細資訊，請參閱 superuser。

SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }

此子句會指定使用者對 Amazon Redshift 系統資料表和檢視的存取層級。

具有 SYSLOG ACCESS 受限制權限的一般使用者只能在使用者可見的系統資料表和檢視中看到該使用者所產生的資料列。預設值為 RESTRICTED。

具有 SYSLOG ACCESS UNCONCTED 權限的一般使用者可以看到使用者可見系統資料表和檢視中的所有資料列，包括其他使用者產生的資料列。UNRESTRICTED 並不會讓一般使用者存取超級使用者可查看的資料表。只有超級使用者可看見超級使用者可查看的資料表。

> **Note**
>
> 若使用者擁有不受限制的系統資料表存取權限，該使用者就能查看其他使用者產生的資料。例如，STL_QUERY 和 STL_QUERYTEXT 包含 INSERT、UPDATE 和 DELETE 陳述式的全文，當中可能包含使用者產生的敏感資料。

所有使用者皆可看到 SVV_TRANSACTIONS 中的所有資料列。

如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

IN GROUP groupname

指定使用者所屬現有群組的名稱。可列出多個群組名稱。

VALID UNTIL abstime

VALID UNTIL 選項會設定絕對時間，在這段時間過後使用者密碼就會失效。根據預設，密碼沒有時間限制。

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字可允許同時連線的最大數目。另外也可能限制每個資料庫的連線數目。如需詳細資訊，請參閱 CREATE DATABASE。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 STV_SESSIONS 系統畫面。

> **Note**
>
> 如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

SESSION TIMEOUT limit

工作階段保持非作用中或閒置的時間上限 (以秒為單位)。範圍是 60 秒 (一分鐘) 到 1,728,000 秒 (20 天)。如果未為使用者設定工作階段逾時，則會套用叢集設定。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制。

當您設定工作階段逾時時，只會套用至新的工作階段。

若要檢視有關作用中使用者工作階段的資訊 (包括開始時間、使用者名稱和工作階段逾時)，請查詢 STV_SESSIONS 系統檢視。若要檢視有關使用者工作階段歷史記錄的資訊，請查詢 STL_SESSIONS 檢視。若要擷取有關資料庫使用者的資訊 (包括工作階段逾時值)，請查詢 SVL_USER_INFO 檢視。

EXTERNALID external_id

與身分提供者相關聯的使用者識別碼。使用者必須停用密碼。如需詳細資訊，請參閱 Amazon Redshift 的原生身分提供者 (IdP) 聯合合。

使用須知

根據預設，所有使用者對 PUBLIC 結構描述都具有 CREATE 和 USAGE 權限。若要不允許使用者在資料庫的 PUBLIC 結構描述內建立物件，請使用 REVOKE 命令來移除該權限。

使用 IAM 身分驗證建立資料庫使用者登入資料時，您可能會希望建立只能使用臨時登入資料登入的超級使用者。您無法停用超級使用者的密碼，但是可以使用隨機產生的 MD5 雜湊字串建立未知的密碼。

```
create user iam_superuser password 'md5A1234567890123456780123456789012' createuser;
```

無論 enable_case_sensitive_identifier 組態選項的設定為何，都會保留以雙引號括住的 username 大小寫。如需詳細資訊，請參閱 enable_case_sensitive_identifier。

範例

以下命令會建立名為 dbuser 的使用者與密碼 "abcD1234"、建立資料庫的權限，以及 30 個連線限制。

```
create user dbuser with password 'abcD1234' createdb connection limit 30;
```

查詢 PG_USER_INFO 目錄資料表以檢視有關資料庫使用者的詳細資訊。

```
select * from pg_user_info;
```

```
 usename   | usesysid | usecreatedb | usesuper | usecatupd | passwd   | valuntil |
 useconfig | useconnlimit
-----------+----------+-------------+----------+-----------+----------+----------
+-----------+-------------
 rdsdb     |        1 | true        | true     | true      | ******** | infinity |
      |
 adminuser |      100 | true        | true     | false     | ******** |          |
      | UNLIMITED
 dbuser    |      102 | true        | false    | false     | ******** |          |
      | 30
```

在以下範例中，帳戶密碼的有效期限至 2017 年 6 月 10 日。

```
create user dbuser with password 'abcD1234' valid until '2017-06-10';
```

以下範例會建立使用者，以及包含特殊字元且區分大小寫的密碼。

```
create user newman with password '@AbC4321!';
```

若要在 MD5 密碼中使用反斜線 ('\')，請使用來源字串中的反斜線逸出反斜線。以下範例會建立名為 slashpass 的使用者，並使用單一反斜線 ('\') 做為密碼。

```
select md5('\\'||'slashpass');

md5
----------------------------------
0c983d1a624280812631c5389e60d48c
```

使用 md5 密碼建立使用者。

```
create user slashpass password 'md50c983d1a624280812631c5389e60d48c';
```

下列範例會建立將閒置工作階段逾時設定為 120 秒的使用者 dbuser。

```
CREATE USER dbuser password 'abcD1234' SESSION TIMEOUT 120;
```

以下範例會建立名為 bob 的使用者。命名空間為 myco_aad。這只是範例。若要成功執行命令，您必須擁有已註冊的身分提供者。如需詳細資訊，請參閱 Amazon Redshift 的原生身分提供者 (IdP) 聯合。

```
CREATE USER myco_aad:bob EXTERNALID "ABC123" PASSWORD DISABLE;
```

CREATE VIEW

在資料庫中建立檢視。檢視不會實際具體化；定義檢視的查詢會在每次查詢中參考檢視時執行。若要建立外部資料表的檢視，請包含 WITH NO SCHEMA BINDING 子句。

若要建立標準檢視，您需要存取基礎資料表或基礎檢視。若要查詢標準檢視，您需要選取檢視本身的許可，但不需要選取基礎資料表的許可。如果您建立參照其他結構描述中資料表或視觀表的視觀表，或是建立參照具體化視觀表的視觀表，則需要使用權限。若要查詢近期繫結視觀表，您需要選取近期繫結視觀表本身的權限。您也應該確認近期繫結檢視的擁有者具有所參考物件 (資料表、檢視或使用者定義的函數) 的選取權限。如需近期繫結視觀表的詳細資訊，請參閱 使用須知。

所需權限

以下是 CREATE VIEW: 所需的權限：

- 對於 CREATE VIEW：
  - 超級使用者
  - 具有 CREATE [OR REPLACE] VIEW 權限的使用者
- 對於 REPLACE VIEW：
  - 超級使用者
  - 具有 CREATE [OR REPLACE] VIEW 權限的使用者
  - 檢視擁有者

語法

```
CREATE [ OR REPLACE ] VIEW name [ ( column_name [, ...] ) ] AS query
[ WITH NO SCHEMA BINDING ]
```

參數

OR REPLACE

如果有同名的檢視存在，則會取代該檢視。您只能將檢視取代為使用相同的資料欄名稱和資料類型產生一組相同資料欄的新查詢。CREATE OR REPLACE VIEW 會鎖定檢視的讀取和寫入，直到操作完成。

當檢視被替換時，其他屬性 (例如所有權和授予的權限) 會保留下來。

name

檢視的名稱。如果有提供結構描述名稱 (例如 myschema.myview)，則會使用指定的結構描述建立檢視。否則，檢視會在目前結構描述中建立。檢視名稱必須與相同結構描述中的任何其他檢視或資料表名稱不同。

若您指定 '#' 開頭的檢視名稱，所建立的檢視會是臨時檢視，而且只可在目前工作階段中看見。

如需有效名稱的相關資訊，請參閱 名稱與識別碼。您無法在系統資料庫 template0、template1 和 padb_harvest 或 sys:internal中建立資料表或視觀表。

column_name

選用的名稱清單，將用於檢視中的資料欄。若未提供任何資料欄名稱，則會從查詢衍生資料欄名稱。單一檢視中可定義的資料欄數目上限為 1,600 個。

query

判斷值為資料表的查詢 (採用 SELECT 陳述式的形式)。此資料表會定義檢視中的資料欄和資料列。

WITH NO SCHEMA BINDING

此子句會指定檢視不會與底層資料庫物件繫結，像是資料表和使用者定義的函數。因此，檢視和其參考的物件之間並無相依性。即使參考的物件不存在，您仍可以建立檢視。由於沒有相依性，您可以捨棄或修改參考物件，而不會影響檢視。Amazon Redshift 在查詢檢視之前不會檢查相依姓。若要檢視近期繫結視觀表的詳細資訊，請執行 PG_GET_LATE_BINDING_VIEW_COLS 函數。

若包含 WITH NO SCHEMA BINDING 子句，則 SELECT 陳述式中參考的資料表和檢視都必須以結構描述名稱限定。檢視建立時，結構描述必須存在，即使參考的資料表不存在也一樣。例如，下列陳述式會傳回錯誤。

```
create view myevent as select eventname from event
with no schema binding;
```

下列陳述式會成功執行。

```
create view myevent as select eventname from public.event
with no schema binding;
```

> **Note**
>
> 您無法在檢視中進行更新、插入或刪除操作。

使用須知

近期繫結檢視

在對檢視進行查詢之前，近期繫結檢視不會檢查底層資料庫物件，像是資料表和其他檢視。因此，您可以修改或捨棄底層物件，而不會捨棄和重新建立檢視。若您捨棄底層物件，對近期繫結檢視的查詢將會失敗。若對近期繫及檢視的查詢參考底層物件中不存在的資料欄，則查詢將會失敗。

若您捨棄後再次建立近期繫結檢視的基礎資料表或檢視，則會建立具有預設存取許可的新物件。您可能需要對將查詢檢視的使用者授予基礎物件的許可。

若要建立近期繫結檢視，請包含 WITH NO SCHEMA BINDING 子句。以下範例會建立不含結構描述繫結的檢視。

```
create view event_vw as select * from public.event
with no schema binding;
```

```
select * from event_vw limit 1;

eventid | venueid | catid | dateid | eventname     | starttime
--------+---------+-------+--------+---------------+--------------------
      2 |     306 |     8 |   2114 | Boris Godunov | 2008-10-15 20:00:00
```

以下範例顯示您可以修改基礎資料表，但不重新建立檢視。

```
alter table event rename column eventname to title;
```

```
select * from event_vw limit 1;

eventid | venueid | catid | dateid | title         | starttime
--------+---------+-------+--------+---------------+--------------------
      2 |     306 |     8 |   2114 | Boris Godunov | 2008-10-15 20:00:00
```

您只能在近期繫結檢視中參考 Amazon Redshift Spectrum 外部資料表。近期繫結檢視的應用方式之一，就是查詢 Amazon Redshift 和 Redshift Spectrum 這兩種資料表。例如，您可以使用 UNLOAD 命令將舊資料封存至 Amazon S3。然後建立參考 Amazon S3 中資料的 Redshift Spectrum 外部資料表，並建立查詢這兩種資料表的檢視。以下範例會使用 UNION ALL 子句聯結 Amazon Redshift SALES 資料表和 Redshift Spectrum SPECTRUM.SALES 資料表。

```
create view sales_vw as
select * from public.sales
union all
select * from spectrum.sales
with no schema binding;
```

如需建立 Redshift Spectrum 外部資料表的相關資訊，包括 SPECTRUM.SALES 資料表，請參閱 開始使用 Amazon Redshift Spectrum。

當您從近期繫結檢視建立標準檢視時，標準檢視的定義會包含進行標準檢視時的近期繫結檢視定義。系統不會追蹤近期繫結檢視的相依性，因此不會在標準檢視中追蹤對近期繫結檢視所做的變更。

若要更新標準檢視表以參考近期繫結檢視的最新定義，請使用您用來建立標準檢視的初始檢視定義來執行 CREATE OR REPLACE VIEW。

請參閱下列範例，了解如何從近期繫結檢視建立標準檢視。

```
create view sales_vw_lbv as
select * from public.sales
with no schema binding;

show view sales_vw_lbv;
                            Show View DDL statement
--------------------------------------------------------------------------------
 create view sales_vw_lbv as select * from public.sales with no schema binding;
(1 row)

create view sales_vw as
select * from sales_vw_lbv;

show view sales_vw;
                                                 Show View DDL statement
-------------------------------------------------------------------------------------------------
 SELECT sales_vw_lbv.price, sales_vw_lbv."region" FROM (SELECT sales.price,
 sales."region" FROM sales) sales_vw_lbv;
(1 row)
```

請注意，標準檢視的 DDL 陳述式中所顯示的近期繫結檢視是在建立標準檢視時定義的，之後不會隨著近期繫結檢視上所做的任何變更進行更新。

範例

範例命令會使用稱為 TICKIT 資料庫的一組物件和資料範例。如需詳細資訊，請參閱 tz 資料庫。

以下命令會從名為 EVENT 的資料表建立名為 myevent 的檢視。

```
create view myevent as select eventname from event
where eventname = 'LeAnn Rimes';
```

以下命令會從名為 USERS 的資料表建立名為 myuser 的檢視。

```
create view myuser as select lastname from users;
```

以下命令會從名為 USERS 的資料表建立或取代名為 myuser 的檢視。

```
create or replace view myuser as select lastname from users;
```

以下範例會建立不含結構描述繫結的檢視。

```
create view myevent as select eventname from public.event
with no schema binding;
```

DEALLOCATE

解除配置預備陳述式。

語法

```
DEALLOCATE [PREPARE] plan_name
```

參數

PREPARE

此關鍵字為選用並且會予以忽略。

plan_name

要解除配置之預備陳述式的名稱。

使用須知

DEALLOCATE 是用來解除配置先前預備的 SQL 陳述式。若您未明確解除配置預備陳述式，則會在目前工作階段結束時將它解除配置。如需預備陳述式的相關資訊，請參閱 PREPARE。

另請參閱

EXECUTE, PREPARE

DECLARE

定義新的資料指標。使用資料指標可從完整查詢的結果集中一次擷取幾個資料列。

擷取資料指標的第一列時，整個結果集會在領導節點上、記憶體中或磁碟上具體化 (如有需要)。由於在大型結果集內使用資料指標可能會對效能造成負面影響，因此建議您盡量使用替代方式。如需詳細資訊，請參閱 使用游標時的效能考量。

您必須在交易區塊內宣告資料指標。每個工作階段中一次只能開啟一個資料指標。

如需詳細資訊，請參閱 FETCH、CLOSE。

語法

```
DECLARE cursor_name CURSOR FOR query
```

參數

cursor_name

新資料指標的名稱。

query

填入資料指標的 SELECT 陳述式。

DECLARE CURSOR 使用須知

若您的用戶端應用程式使用 ODBC 連線，且您的查詢建立的結果集太大，無法納入記憶體中，則您可以使用資料指標將結果集串流到用戶端應用程式。當您使用資料指標時，整個結果集會在領導節點上具體化，然後您的用戶端就能以遞增方式擷取結果。

> **Note**
>
> 若要在 Microsoft Windows 的 ODBC 中啟用資料指標，請在您用於 Amazon Redshift 的 ODBC DSN 中啟用使用宣告/擷取選項。建議您使用 ODBC DSN 選項對話方塊中的 Cache Size (快取大小) 欄位，將多節點叢集上的 ODBC 快取大小設定為 4,000 或更大的數字，以減少往返次數。在單一節點叢集上，將 Cache Size (快取大小) 設定為 1,000。

由於使用資料指標可能會對效能造成負面影響，因此建議您盡量使用替代方式。如需詳細資訊，請參閱使用游標時的效能考量。

在滿足以下限制的情況下，可支援 Amazon Redshift 資料指標：

- 每個工作階段中一次只能開啟一個資料指標。
- 資料指標必須在交易內使用 (BEGIN … END)。
- 所有資料指標累積的結果集大小上限受限於叢集節點類型。若您需要更大的結果集，可將大小調整為 XL 或 8XL 節點組態。

  如需詳細資訊，請參閱 游標限制條件。

游標限制條件

擷取資料指標的第一列時，整個結果集會在領導節點上具體化。若結果集無法納入記憶體內，則會視需要寫入磁碟中。為保護領導節點的完整性，Amazon Redshift 會根據叢集的節點類型對所有資料指標結果集的大小強制執行限制條件。

下表說明每個叢集節點類型的結果集大小總計上限。結果集大小上限的單位是 MB。

節點類型	每個叢集的結果集上限 (MB)
RA3 16XL 多個節點	14400000
DC2 Large 單節點	8000
DC2 Large 多個節點	192000
DC2 8XL 多個節點	3200000
RA3 4XL 多個節點	3200000

節點類型	每個叢集的結果集上限 (MB)
RA3 XLPLUS 多個節點	1000000
RA3 XLPLUS 單節點	64000
Amazon Redshift Serverless	150000

若要檢視叢集的使用中資料指標組態，請以超級使用者身分查詢 STV_CURSOR_CONFIGURATION 系統資料表。若要檢視使用中資料指標的狀態，請查詢 STV_ACTIVE_CURSORS 系統資料表。使用者只能看見自己擁有之資料指標的資料列，但超級使用者可檢視所有資料指標。

使用游標時的效能考量

由於資料指標會在開始傳回結果至用戶端之前，於領導節點上將整個結果集具體化，因此在非常大型的結果集內使用資料指標會對效能產生負面影響。強烈建議您不要在非常大型的結果集內使用資料指標。在某些情況下，例如應用程式使用 ODBC 連線時，資料指標可能會是唯一可行的解決方案。不過我們建議您盡量使用以下替代方式：

- 使用 UNLOAD 匯出大型資料表。使用 UNLOAD 時，運算節點會平行運作，將資料直接傳輸至 Amazon Simple Storage Service 上的資料檔案。如需詳細資訊，請參閱 卸載資料。
- 在您的用戶端應用程式中設定 JDBC 擷取大小參數。如果您使用 JDBC 連線，而且遇到用戶端 out-of-memory 錯誤，您可以設定 JDBC 擷取大小參數，讓用戶端能夠以較小的批次擷取結果集。如需詳細資訊，請參閱 設定 JDBC 擷取大小參數。

DECLARE CURSOR 範例

以下範例會宣告名為 LOLLAPALOOZA 的資料指標來選取 Lollapalooza 活動的銷售資訊，然後使用資料指標從結果集擷取資料列：

```
-- Begin a transaction

begin;

-- Declare a cursor

declare lollapalooza cursor for
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
```

```
where sales.eventid = event.eventid
and eventname='Lollapalooza';

-- Fetch the first 5 rows in the cursor lollapalooza:

fetch forward 5 from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-05-01 19:00:00 |   92.00000000 |       3
 Lollapalooza | 2008-11-15 15:00:00 |  222.00000000 |       2
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       3
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       4
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       1
(5 rows)

-- Fetch the next row:

fetch next from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-10-06 14:00:00 |  114.00000000 |       2

-- Close the cursor and end the transaction:

close lollapalooza;
commit;
```

下列範例會使用資料表中的所有結果在參考游標上循環：

```
CREATE TABLE tbl_1 (a int, b int);
INSERT INTO tbl_1 values (1, 2),(3, 4);

CREATE OR REPLACE PROCEDURE sp_cursor_loop() AS $$
DECLARE
    target record;
    curs1 cursor for select * from tbl_1;
BEGIN
    OPEN curs1;
    LOOP
        fetch curs1 into target;
        exit when not found;
```

```
        RAISE INFO 'a %', target.a;
    END LOOP;
    CLOSE curs1;
END;
$$ LANGUAGE plpgsql;

CALL sp_cursor_loop();

SELECT message
   from svl_stored_proc_messages
   where querytxt like 'CALL sp_cursor_loop()%';

  message
----------
      a 1
      a 3
```

DELETE

刪除資料表中的資料列。

> **Note**
>
> 單一 SQL 陳述式的大小上限為 16 MB。

語法

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
DELETE [ FROM ] { table_name | materialized_view_name }
    [ { USING } table_name, ... ]
    [ WHERE condition ]
```

參數

WITH 子句

指定一個或多個 common-table-expressions 的選用子句。請參閱WITH 子句。

FROM

FROM 關鍵字為選用，但指定了 USING 子句時除外。`delete from event;` 和 `delete event;` 這兩個陳述式是相同的操作，會將 EVENT 資料表的所有資料列移除。

> **Note**
>
> 若要刪除資料表中的所有資料列，請對資料表執行 TRUNCATE。TRUNCATE 比 DELETE 更有效率，而且不需要 VACUUM 和 ANALYZE。不過請注意，TRUNCATE 會遞交其執行所在的交易。

table_name

暫時性或持久性資料表。只有資料表的擁有者，或具有資料表 DELETE 權限的使用者可從資料表中刪除資料列。

請考慮使用 TRUNCATE 命令在大型資料表上快速執行非限定的刪除操作；請參閱 TRUNCATE。

> **Note**
>
> 從資料表中刪除大量資料列之後：
>
> - 清空資料表以回收儲存空間和重新排序資料列。
> - 分析資料表以更新查詢規劃器的統計資訊。

materialized_view_name

具體化視觀表。DELETE 陳述式可在用於 串流擷取 的具體化視觀表上作用。只有具體化視觀表的擁有者或在具體化視觀表上具有 DELETE 權限的使用者，才能從中刪除資料列。

您無法使用未授予使用者 IGNORE RLS 權限的資料列層級安全性 (RLS) 政策，在具體化視觀表上執行 DELETE 來進行串流擷取。有一個例外情況：如果執行 DELETE 的使用者已授予 IGNORE RLS，則可成功執行。如需詳細資訊，請參閱 RLS 政策擁有權和管理。

USING table_name, ...

USING 關鍵字會在 WHERE 子句條件中參考其他資料表時，用來加入資料表清單。例如，以下陳述式會從 EVENT 資料表中，刪除所有滿足 EVENT 和 SALES 資料表聯結條件的資料列。SALES 資料表必須在 FROM 清單中明確命名：

```
delete from event using sales where event.eventid=sales.eventid;
```

若您在 USING 子句中重複目標資料表名稱，則 DELETE 操作會執行自我聯結。您可以在 WHERE 子句中使用子查詢來取代 USING 語法，做為撰寫相同查詢的替代方式。

WHERE condition

此選用子句會限制僅刪除符合條件的資料列。例如，條件可以是資料欄上的限制、聯結條件，或根據查詢結果的條件。查詢可參考 DELETE 命令的目標以外的資料表。例如：

```
delete from t1
where col1 in(select col2 from t2);
```

如未指定任何條件，則會刪除資料表中的所有資料列。

範例

從 CATEGORY 資料表刪除所有資料列：

```
delete from category;
```

從 CATEGORY 資料表刪除 CATID 值介於 0 和 9 之間的資料列：

```
delete from category
where catid between 0 and 9;
```

從 LISTING 資料表刪除其 SELLERID 值不存在 SALES 資料表中的資料列：

```
delete from listing
where listing.sellerid not in(select sales.sellerid from sales);
```

以下兩個查詢都會根據 EVENT 資料表的聯結和對 CATID 資料欄的額外限制，從 CATEGORY 資料表刪除一個資料列：

```
delete from category
using event
where event.catid=category.catid and category.catid=9;
```

```
delete from category
```

```
where catid in
(select category.catid from category, event
where category.catid=event.catid and category.catid=9);
```

下列查詢會刪除 mv_cities 具體化視觀表中的所有資料列。此範例中的具體化視觀表名稱為範例：

```
delete from mv_cities;
```

DESC DATASHARE

顯示使用 ALTER DATASHARE 新增至資料共用中的資料庫物件清單。Amazon Redshift 會顯示資料表、檢視和函數的名稱、資料庫、結構描述及類型。

您可以使用系統檢視來找到有關資料清單物件的其他資訊。如需詳細資訊，請參閱 SVV 資料庫物件和 SVV 資料庫。

語法

```
DESC DATASHARE datashare_name [ OF [ ACCOUNT account_id ] NAMESPACE namespace_guid ]
```

參數

datashare_name

資料共用的名稱。

NAMESPACE namespace_guid

指定資料共用所用命名空間的值。當您以取用者叢集管理員身分執行 DESC DATAHSARE 時，請指定 NAMESPACE 參數以檢視輸入資料共用。

ACCOUNT account_id

指定資料共用所屬帳戶的值。

使用須知

身為消費者帳戶管理員，當您執行 DESC DATASHARE 以查看 AWS 帳戶內的輸入資料庫時，請指定「命名空間」選項。當您執行「描述資料清理」以查看跨 AWS 帳戶的輸入資料庫時，請指定「帳戶」和「命名空間」選項。

範例

下列範例顯示生產者叢集上有關輸出資料共用的資訊。

```
DESC DATASHARE salesshare;

producer_account |          producer_namespace           | share_type  | share_name   |
 object_type |        object_name           |  include_new
-----------------+--------------------------------------+-------------+--------------
+-------------+------------------------------+--------------
 123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d  | OUTBOUND    |  salesshare  |
 TABLE       | public.tickit_sales_redshift |
 123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d  | OUTBOUND    |  salesshare  |
 SCHEMA      | public                       |   t
```

下列範例顯示取用者叢集上有關輸入資料共用的資訊。

```
DESC DATASHARE salesshare of ACCOUNT '123456789012' NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';

 producer_account |          producer_namespace          | share_type | share_name |
 object_type |         object_name          |  include_new
-----------------+--------------------------------------+------------+------------
+-------------+------------------------------+--------------
 123456789012     | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare |
 table       | public.tickit_sales_redshift |
 123456789012     | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare |
 schema      | public                       |
(2 rows)
```

DESC IDENTITY PROVIDER

顯示身分提供者的相關資訊。只有超級使用者可以描述身分提供者。

語法

```
DESC IDENTITY PROVIDER identity_provider_name
```

參數

identity_provider_name

身分提供者的名稱。

範例

下列範例顯示身分提供者的相關資訊。

```
DESC IDENTITY PROVIDER azure_idp;
```

輸出範例。

```
  uid   |   name    | type  |             instanceid              | namespc |
                                          params
                           | enabled
--------+-----------+-------+-------------------------------------+---------
+-----------------------------------------------------------------------------------------------------------------
+---------
 126692 | azure_idp | azure | e40d4bb2-7670-44ae-bfb8-5db013221d73 | aad     |
 {"issuer":"https://login.microsoftonline.com/e40d4bb2-7670-44ae-bfb8-5db013221d73/
v2.0", "client_id":"871c010f-5e61-4fb1-83ac-98610a7e9110", "client_secret":'',
 "audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift", "https://
analysis.windows.net/powerbi/connector/AWSRDS"]} | t
(1 row)
```

DETACH MASKING POLICY

從資料欄中分離已附加的動態資料遮罩政策。如需動態資料遮罩的相關資訊，請參閱 動態資料遮罩。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以分離遮罩政策。

語法

```
DETACH MASKING POLICY policy_name
   ON { table_name }
   ( output_column_names )
   FROM { user_name | ROLE role_name | PUBLIC };
```

參數

policy_name

欲分離的遮罩政策名稱。

table_name

要將遮罩政策分離的資料表名稱。

output_column_names

已附加遮罩政策的資料欄名稱。

user_name

已附加遮罩政策的使用者名稱。

您只能在單一 DETACH MASKING POLICY 陳述式中設定 user_name、role_name 和 PUBLIC 中的其中一個。

role_name

已附加遮罩政策的角色名稱。

您只能在單一 DETACH MASKING POLICY 陳述式中設定 user_name、role_name 和 PUBLIC 中的其中一個。

PUBLIC

顯示政策已附加至資料表中的所有使用者。

您只能在單一 DETACH MASKING POLICY 陳述式中設定 user_name、role_name 和 PUBLIC 中的其中一個。

DETACH RLS POLICY

將資料表上的資料列層級安全性政策從一或多個使用者或角色中分離。

超級使用者和具有 `sys:secadmin` 角色的使用者或角色可以分離政策。

語法

```
DETACH RLS POLICY policy_name ON [TABLE] table_name [, ...]
```

```
FROM { user_name | ROLE role_name | PUBLIC } [, ...]
```

參數

policy_name

政策的名稱。

ON [TABLE] table_name [, ...]

分離資料列層級安全性政策的資料表或檢視。

FROM { user_name | ROLE role_name | PUBLIC} [, ...]

指定政策是否要從一或多個指定的使用者或角色中分離。

使用須知

使用 DETACH RLS POLICY 陳述式時，請注意以下內容：

- 您可以從關係、使用者、角色或公用項目中分離政策。

範例

下列範例會將資料表上的政策與角色分離。

```
DETACH RLS POLICY policy_concerts ON tickit_category_redshift FROM ROLE analyst, ROLE
 dbadmin;
```

DROP DATABASE

捨棄資料庫。

您無法在交易區塊 (BEGIN ... END) 內執行 DROP DATABASE。如需交易的相關資訊，請參閱 可序列化隔離。

語法

```
DROP DATABASE database_name
```

參數

database_name

要捨棄之資料庫的名稱。您無法捨棄 dev、padb_harvest、template0、template1 或 sys:internal 資料庫，也無法捨棄目前的資料庫。

若要捨棄外部資料庫，請捨棄外部結構描述。如需詳細資訊，請參閱 DROP SCHEMA。

DROP DATABASE 使用須知

使用 DROP DATABASE 陳述式時，請考慮下列事項：

- 一般而言，我們建議您不要使用 DROP AWS Data Exchange DATABASE 陳述式刪除包含資料追蹤的資料庫。如果 AWS 帳戶 這樣做，則可以訪問數據的訪問權限將失去訪問權限。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。

  下列範例顯示當包含資料清理的資料庫遭到捨棄時發生錯誤。 AWS Data Exchange

  ```
  DROP DATABASE test_db;
  ERROR:   Drop of database test_db that contains ADX-managed datashare(s)
   requires session variable datashare_break_glass_session_var to be set to value
   'ce8d280c10ad41'
  ```

  若要允許捨棄資料庫，請設定下列變數，然後再次執行 DROP DATABASE 陳述式。

  ```
  SET datashare_break_glass_session_var to 'ce8d280c10ad41';
  ```

  ```
  DROP DATABASE test_db;
  ```

  在這種情況下，Amazon Redshift 會產生隨機的一次性值來設定工作階段變數，以允許對包含 AWS Data Exchange 資料共用的資料庫執行 DROP DATABASE。

範例

以下範例會捨棄名為 TICKIT_TEST 的資料庫：

```
drop database tickit_test;
```

DROP DATASHARE

捨棄資料共用。此命令無法還原。

只有超級使用者或資料共用擁有者可以捨棄資料共用。

所需權限

以下是 DROP DATASHARE 所需的權限：

- 超級使用者
- 具有 DROP DATASHARE 權限的使用者
- 資料共用擁有者

語法

```
DROP DATASHARE datashare_name;
```

參數

datashare_name

要捨棄的資料共用名稱。

DROP DATASHARE 使用須知

使用 DROP DATASHARE 陳述式時，請考慮下列事項：

- 一般而言，我們建議您不要使用 DROP AWS Data Exchange DATASHARE 陳述式卸除資料記憶體。如果 AWS 帳戶 這樣做，則可以訪問數據的訪問權限將失去訪問權限。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。

  下列範例顯示捨棄 AWS Data Exchange 資料命令時發生錯誤。

  ```
  DROP DATASHARE salesshare;
  ERROR:  Drop of ADX-managed datashare salesshare requires session variable
   datashare_break_glass_session_var to be set to value '620c871f890c49'
  ```

若要允許刪除 AWS Data Exchange 資料清理，請設定下列變數，然後再次執行 DROP DATASHARE 陳述式。

```
SET datashare_break_glass_session_var to '620c871f890c49';
```

```
DROP DATASHARE salesshare;
```

在這種情況下，Amazon Redshift 會產生一個隨機的一次性值，以設定工作階段變數，以允許資料追蹤的 DROP 資料清理。 AWS Data Exchange

範例

以下範例會捨棄名為 `salesshare` 的資料共用。

```
DROP DATASHARE salesshare;
```

DROP EXTERNAL VIEW (預覽)

這是適用於 Amazon Redshift 的資料目錄中的發行前版本文件檢視，屬於預覽版本。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，網址為 https://console.aws.amazon.com/redshiftv2/。
2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。 AWS 區域 會列出目前帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。

4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的建立叢集。
5. 選擇建立叢集按鈕以建立預覽叢集。

> **Note**
>
> `preview_2023` 軌跡是最近可用的預覽軌跡。此軌跡僅支援使用 RA3 節點類型建立叢集。不支援節點類型 DC2 和任何較舊的節點類型。

6. 當您的預覽叢集可用時，請使用 SQL 用戶端載入和查詢資料。

Data Catalog 視觀表預覽功能僅適用於以下區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

您也可以建立預覽工作群組來測試 Data Catalog 視觀表。您無法在生產環境中使用這些功能，也無法將工作群組移至另一個工作群組。如需了解預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。如需如何建立預覽工作群組的指示，請參閱 https://docs.aws.amazon.com/redshift/latest/mgmt/serverless-workgroup-preview.html。

從資料庫中捨棄外部檢視。捨棄外部檢視後會將其從檢視相關聯的所有 SQL 引擎中移除，例如 Amazon Athena 和 Amazon EMR Spark。此命令無法反轉。如需有關 Data Catalog 視觀表的詳細資訊，請參閱建立 Data Catalog 視觀表 (預覽)。

語法

```
DROP EXTERNAL VIEW schema_name.view_name [ IF EXISTS ]
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
 external_schema_name.view_name}
```

參數

schema_name.view_name

附加到 AWS Glue 資料庫的結構描述，後面接著檢視的名稱。

IF EXISTS

只有在檢視存在時才捨棄檢視。

catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name | external_schema_name.view_name

捨棄視觀表時所使用的結構描述標記法。您可以指定使用 AWS Glue Data Catalog您建立的 Glue 資料庫或您建立的外部結構描述。如需詳細資訊，請參閱CREATE DATABASE 和 CREATE EXTERNAL SCHEMA。

query_definition

Amazon Redshift 執行以變更檢視的 SQL 查詢的定義。

範例

下列範例會捨棄名為 sample_schema.glue_data_catalog_view 的 Data Catalog 視觀表。

```
DROP EXTERNAL VIEW sample_schema.glue_data_catalog_view IF EXISTS
```

DROP FUNCTION

從資料庫移除使用者定義的函數 (UDF)。必須指定函數的簽章 (或引數資料類型的清單)，因為可能有多個同名但簽章不同的函數存在。您無法捨棄 Amazon Redshift 內建函數。

此命令無法還原。

所需權限

以下是 DROP FUNCTION 所需的權限：

- 超級使用者
- 具有 DROP FUNCTION 權限的使用者
- 函數擁有者

語法

```
DROP FUNCTION name
( [arg_name] arg_type   [, ...] )
[ CASCADE | RESTRICT ]
```

參數

name

要移除的函數名稱。

arg_name

輸入引數的名稱。DROP FUNCTION 會忽略引數名稱，因為只需要引數資料類型即可判斷函數的身分。

arg_type

輸入引數的資料類型。您可以提供最多包含 32 種資料類型的逗號分隔清單。

CASCADE

此關鍵字指定自動捨棄取決於函數的物件，例如檢視。

若要建立不相依於函數的檢視，請在檢視定義中包含 WITH NO SCHEMA BINDING 子句。如需詳細資訊，請參閱 CREATE VIEW。

RESTRICT

此關鍵字指定，若有任何物件取決於函數，則不捨棄函數並傳回訊息。這是預設動作。

範例

下列範例會捨棄名為 `f_sqrt` 的函數：

```
drop function f_sqrt(int);
```

若要移除有相依性的函數，請使用 CASCADE 選項，如下所範例所示：

```
drop function f_sqrt(int)cascade;
```

DROP GROUP

刪除使用者群組。此命令無法還原。此命令不會刪除群組中的個別使用者。

請參閱 DROP USER 以刪除個別使用者。

語法

```
DROP GROUP name
```

參數

name

要刪除的使用者群組名稱。

範例

下列範例會刪除使guests用者群組：

```
DROP GROUP guests;
```

如果群組有物件上的任何權限，則您無法捨棄群組。如果您嘗試捨棄這類群組，則會收到下列錯誤。

```
ERROR: group "guests" can't be dropped because the group has a privilege on some object
```

如果群組擁有物件的權限，您必須先撤銷權限，才能卸除群組。若要尋找guests群組有權限的物件，請使用下列範例。如需有關範例中使用之中繼資料檢視的詳細資訊，請參閱 SVV _ 關係權限。

```
SELECT DISTINCT namespace_name, relation_name, identity_name, identity_type
FROM svv_relation_privileges
WHERE identity_type='group' AND identity_name='guests';

+----------------+---------------+---------------+---------------+
| namespace_name | relation_name | identity_name | identity_type |
+----------------+---------------+---------------+---------------+
| public         | table1        | guests        | group         |
+----------------+---------------+---------------+---------------+
| public         | table2        | guests        | group         |
+----------------+---------------+---------------+---------------+
```

下列範例會撤銷 public 使用者群組的 guests 結構描述中所有資料表的所有權限，然後捨棄群組。

```
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM GROUP guests;
DROP GROUP guests;
```

DROP IDENTITY PROVIDER

刪除身分提供者。此命令無法還原。只有超級使用者可以捨棄身分提供者。

語法

```
DROP IDENTITY PROVIDER identity_provider_name [ CASCADE ]
```

參數

identity_provider_name

要刪除的身分提供者名稱。

CASCADE

刪除身分提供者時，刪除附加至身分提供者的使用者和角色。

範例

下列範例會刪除 oauth_provider 身分提供者。

```
DROP IDENTITY PROVIDER oauth_provider;
```

如果您捨棄身分提供者，某些使用者可能無法登入，或使用設定為使用身分提供者的用戶端工具。

DROP LIBRARY

從資料庫移除自訂 Python 程式庫。只有程式庫擁有者或超級使用者可以捨棄程式庫。

DROP LIBRARY 無法在交易區塊內 (BEGIN … END) 執行。如需交易的相關資訊，請參閱 可序列化隔離。

此命令無法還原。DROP LIBRARY 命令會立即遞交。如果倚賴程式庫的 UDF 同時執行，UDF 可能會失敗，即使 UDF 是在交易內執行也一樣。

如需詳細資訊，請參閱 CREATE LIBRARY。

所需權限

以下是 DROPLIBRARY 所需的權限：

- 超級使用者
- 具有 DROP LIBRARY 權限的使用者
- 程式庫擁有者

語法

```
DROP LIBRARY library_name
```

參數

library_name

程式庫的名稱。

DROP MASKING POLICY

從所有資料庫中捨棄動態資料遮罩政策。您無法捨棄仍然附加至一或多個資料表的遮罩政策。如需動態資料遮罩的相關資訊，請參閱 動態資料遮罩。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以捨棄遮罩政策。

語法

```
DROP MASKING POLICY policy_name;
```

參數

policy_name

要捨棄的遮罩政策名稱。

DROP MODEL

從資料庫移除模型。只有模型擁有者或超級使用者可以捨棄模型。

DROP MODEL 也會刪除從此模型衍生的所有相關聯預測函數、與該模型相關的所有 Amazon Redshift 成品，以及與模型相關的所有 Amazon S3 資料。雖然模型仍在 Amazon 中進行訓練 SageMaker，但 DROP MODEL 將取消這些操作。

此命令無法還原。DROP MODEL 命令會立即遞交。

所需的許可

以下是 DROP MODEL 所需的許可：

- 超級使用者
- 具有 DROP MODEL 許可的使用者
- 模型擁有者
- 結構描述擁有者

語法

```
DROP MODEL [ IF EXISTS ] model_name
```

參數

IF EXISTS

此子句會指出，若指定的結構描述已存在，則命令不應進行任何變更，且應傳回結構描述存在的訊息。

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

範例

下列範例會捨棄模型 demo_ml.customer_churn。

```
DROP MODEL demo_ml.customer_churn
```

DROP MATERIALIZED VIEW

移除具體化檢視。

如需具體化檢視的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

語法

```
DROP MATERIALIZED VIEW [ IF EXISTS ] mv_name [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

指定要檢查具名具體化檢視是否存在的子句。如果具體化檢視不存在，則 DROP MATERIALIZED VIEW 命令會傳回錯誤訊息。這個子句在編寫指令碼時相當有用，可用來避免指令碼在您卸除不存在的具體化檢視時失敗。

mv_name

要卸除的具體化檢視名稱。

CASCADE

指示要自動刪除具體化視觀表相依物件的子句，例如其他視觀表。

RESTRICT

指出如果有任何物件相依，就不刪除具體化視觀表的子句。此為預設值。

使用須知

只有具體化檢視的擁有者可以在該檢視上使用 DROP MATERIALIZED VIEW。超級使用者或特別被授予 DROP 權限的使用者可以是此狀況的例外。

當您為具體化視觀表撰寫捨棄陳述式，且存在名稱相符的檢視時，這會導致錯誤，並指示您使用 DROP VIEW。即使在您使用 DROP MATERIALIZED VIEW IF EXISTS 的情況下，也會發生錯誤。

範例

以下範例會卸除 tickets_mv 具體化檢視。

```
DROP MATERIALIZED VIEW tickets_mv;
```

DROP PROCEDURE

捨棄程序。若要捨棄程序，程序名稱和輸入引數資料類型 (簽章) 都是必要的。您可以選擇性包含完整引數資料類型，包括 OUT 引數。若要尋找程序的簽章，請使用 SHOW PROCEDURE 命令。如需程序簽章的相關資訊，請參閱 PG_PROC_INFO。

所需權限

以下是 DROP PROCEDURE 所需的權限：

- 超級使用者
- 具有 DROP PROCEDURE 權限的使用者
- 程序擁有者

語法

```
DROP PROCEDURE sp_name ( [ [ argname ] [ argmode ] argtype [, ...] ] )
```

參數

sp_name

要移除的程序的名稱。

argname

輸入引數的名稱。DROP PROCEDURE 會忽略引數名稱，因為只需要引數資料類型即可判斷程序的身分。

argmode

引數的模式，可以是 IN、OUT 或 INOUT。OUT 引數是選用的，因為不用來識別預存程序。

argtype

輸入引數的資料類型。如需支援的資料類型清單，請參閱 資料類型。

範例

下列範例捨棄名為 `quarterly_revenue` 的預存程序。

```
DROP PROCEDURE quarterly_revenue(volume INOUT bigint, at_price IN numeric,result OUT
 int);
```

DROP RLS POLICY

捨棄所有資料庫中所有資料表的資料列層級安全性政策。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以捨棄政策。

語法

```
DROP RLS POLICY [ IF EXISTS ] policy_name [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

指出指定政策是否已存在的子句。

policy_name

政策的名稱。

CASCADE

指示在捨棄政策之前，自動將政策從所有附加的資料表分離的子句。

RESTRICT

指出政策在附加至某些資料表時不要捨棄政策。此為預設值。

範例

下列範例會捨棄資料列層級安全性政策。

```
DROP RLS POLICY policy_concerts;
```

DROP ROLE

從資料庫中移除角色。只有建立角色的角色擁有者、具有 WITH ADMIN 選項的使用者或超級使用者可以捨棄角色。

如果角色已授予使用者或相依於此角色的其他角色，則您無法捨棄角色。

所需權限

以下是 DROP ROLE 所需的權限：

- 超級使用者
- 角色擁有者，也就是建立角色的使用者，或是已被授予角色且該角色具有 WITH ADMIN OPTION 權限的使用者。

語法

```
DROP ROLE role_name [ FORCE | RESTRICT ]
```

參數

role_name

角色的名稱。

[FORCE | RESTRICT]

預設設定為 RESTRICT。當您嘗試捨棄繼承另一個角色的角色時，Amazon Redshift 會擲回錯誤。使用 FORCE 移除所有角色指派 (如果有的話)。

範例

下列範例會捨棄 sample_role 角色。

```
DROP ROLE sample_role FORCE;
```

下列範例會嘗試捨棄角色 sample_role1，該角色已透過預設 RESTRICT 選項授予使用者。

```
CREATE ROLE sample_role1;
GRANT sample_role1 TO user1;
DROP ROLE sample_role1;
ERROR:  cannot drop this role since it has been granted on a user
```

若要成功捨棄已授予使用者的 sample_role1，請使用 FORCE 選項。

```
DROP ROLE sample_role1 FORCE;
```

下列範例會嘗試捨棄角色 sample_role2，該角色已透過預設 RESTRICT 選項，讓另一個角色與其相依。

```
CREATE ROLE sample_role1;
CREATE ROLE sample_role2;
GRANT sample_role1 TO sample_role2;
DROP ROLE sample_role2;
ERROR:  cannot drop this role since it depends on another role
```

若要成功捨棄具有其他相依角色的 sample_role2，請使用 FORCE 選項。

```
DROP ROLE sample_role2 FORCE;
```

DROP SCHEMA

刪除結構描述。針對外部結構描述，您也可以捨棄與結構描述相關聯的外部資料庫。此命令無法還原。

所需權限

以下是 DROP SCHEMA 所需的權限：

- 超級使用者
- 結構描述擁有者
- 具有 DROP SCHEMA 權限的使用者

語法

```
DROP SCHEMA [ IF EXISTS ] name [, ...]
[ DROP EXTERNAL DATABASE ]
[ CASCADE | RESTRICT ]
```

參數

IF EXISTS

此子句會指出，若指定的結構描述不存在，則命令不應進行任何變更，且應傳回結構描述不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在編寫指令碼時很實用，如此指令碼就不會因為 DROP SCHEMA 對不存在的結構描述執行而失敗。

name

要捨棄的結構描述名稱。您可以指定多個結構描述名稱，以逗號分隔。

DROP EXTERNAL DATABASE

此子句指出是否捨棄外部結構描述，如果存在則捨棄與外部結構描述相關聯的外部資料庫。如果外部資料庫不存在，此命令會傳回訊息，說明外部資料庫不存在。如果已捨棄多個外部資料庫，則會捨棄與指定結構描述相關聯的所有資料庫。

如果外部資料庫包含相依物件 (例如資料表)，請併入 CASCADE 選項以同時捨棄相依物件。

捨棄外部資料庫時，也會捨棄資料庫中與資料庫相關聯的任何其他外部結構描述。也會捨棄在使用資料庫的其他外部結構描述中所定義的資料表。

DROP EXTERNAL DATABASE 不支援儲存在 HIVE 中繼存放區的外部資料庫。

CASCADE

此關鍵字指出自動捨棄結構描述中的所有物件。如果指定 DROP EXTERNAL DATABASE，則也會捨棄外部資料庫中的所有物件。

RESTRICT

此關鍵字指出，若結構描述或外部資料庫含任何物件則不捨棄。這是預設動作。

範例

以下範例會刪除名為 S_SALES 的結構描述。此範例使用 RESTRICT 做為安全機制，如此一來，若結構描述包含任何物件，就不會將其刪除。在此情況下，您需要先刪除結構描述物件，再刪除結構描述。

```
drop schema s_sales restrict;
```

下列範例會刪除名為 S_SALES 的結構描述，以及相依於該結構描述的所有物件。

```
drop schema s_sales cascade;
```

下列範例會捨棄 S_SALES 結構描述 (如存在的話)，或不執行任何動作，並於結構描述不存在時傳回訊息。

```
drop schema if exists s_sales;
```

以下範例會刪除名為 S_SPECTRUM 的外部結構描述，並建立與其相關聯的外部資料庫。此範例使用 RESTRICT，如此一來，若結構描述和資料庫包含任何物件，就不會將其刪除。在此情況下，您會需要先刪除相依物件，再刪除結構描述和資料庫。

```
drop schema s_spectrum drop external database restrict;
```

以下範例會刪除名為多個結構描述和與其相關聯的外部資料庫，以及任何相依物件。

```
drop schema s_sales, s_profit, s_revenue drop external database cascade;
```

DROP TABLE

從資料庫移除資料表。

如果您要嘗試清空資料表中的資料列，但不移除資料表，請使用 DELETE 或 TRUNCATE 命令。

DROP TABLE 會移除目標資料表上存在的限制條件。使用單一 DROP TABLE 命令即可移除多個資料表。

您無法在交易內 (BEGIN … END) 對外部資料表執行 DROP TABLE。如需交易的相關資訊，請參閱 可序列化隔離。

若要尋找將 DROP 權限授予群組的範例，請參閱 GRANT 範例。

所需權限

以下是 DROP TABLE 所需的權限：

- 超級使用者
- 具有 DROP TABLE 權限的使用者
- 具有結構描述的 USAGE 權限的資料表擁有者

語法

```
DROP TABLE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

此子句會指出，若指定的資料表不存在，則命令不應進行任何變更，且應傳回資料表不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在編寫指令碼時很實用，如此指令碼就不會因為 DROP TABLE 對不存在的資料表執行而失敗。

name

要捨棄的資料表名稱。

CASCADE

此子句指出，自動捨棄取決於資料表的物件，例如檢視。

若要建立不相依於其他資料庫物件的檢視，例如檢視和資料表，請在檢視定義中包含 WITH NO SCHEMA BINDING 子句。如需詳細資訊，請參閱 CREATE VIEW。

RESTRICT

此子句指出，若有任何物件相依於資料表，則不捨棄資料表。這是預設動作。

範例

丟棄無相依性的資料表

以下範例會建立並捨棄名為 FEEDBACK 且沒有相依性的資料表：

```
create table feedback(a int);

drop table feedback;
```

若資料表包含的資料欄為檢視或其他資料表所參考，Amazon Redshift 會顯示以下訊息。

```
Invalid operation: cannot drop table feedback because other objects depend on it
```

同步丟棄兩個資料表

以下命令集會建立 FEEDBACK 資料表和 BUYERS 資料表，然後利用單一命令捨棄這兩個資料表：

```
create table feedback(a int);

create table buyers(a int);

drop table feedback, buyers;
```

丟棄具有相依性的資料表

以下步驟顯示如何使用 CASCADE 參數捨棄名為 FEEDBACK 的資料表。

首先，使用 CREATE TABLE 命令建立名為 FEEDBACK 的簡單資料表：

```
create table feedback(a int);
```

接著使用 CREATE VIEW 命令建立名為 FEEDBACK_VIEW 的檢視，此檢視相依於 FEEDBACK 資料表：

```
create view feedback_view as select * from feedback;
```

以下範例會捨棄 FEEDBACK 資料表，並且也會捨棄 FEEDBACK_VIEW 檢視，因為 FEEDBACK_VIEW 相依於 FEEDBACK 資料表：

```
drop table feedback cascade;
```

檢視資料表的相依性

若要傳回資料表的相依性，請使用下列範例。將 *my_schema* 和 *my_table* 取代為您自己的結構描述和資料表。

```
SELECT dependent_ns.nspname as dependent_schema
, dependent_view.relname as dependent_view
, source_ns.nspname as source_schema
, source_table.relname as source_table
, pg_attribute.attname as column_name
FROM pg_depend
JOIN pg_rewrite ON pg_depend.objid = pg_rewrite.oid
JOIN pg_class as dependent_view ON pg_rewrite.ev_class = dependent_view.oid
JOIN pg_class as source_table ON pg_depend.refobjid = source_table.oid
JOIN pg_attribute ON pg_depend.refobjid = pg_attribute.attrelid
    AND pg_depend.refobjsubid = pg_attribute.attnum
JOIN pg_namespace dependent_ns ON dependent_ns.oid = dependent_view.relnamespace
```

```
JOIN pg_namespace source_ns ON source_ns.oid = source_table.relnamespace
WHERE
source_ns.nspname = 'my_schema'
AND source_table.relname = 'my_table'
AND pg_attribute.attnum > 0
ORDER BY 1,2
LIMIT 10;
```

若要捨棄 *my_table* 及其相依性，請使用以下範例。此範例也會傳回已捨棄資料表的所有相依性。

```
DROP TABLE my_table CASCADE;

SELECT dependent_ns.nspname as dependent_schema
, dependent_view.relname as dependent_view
, source_ns.nspname as source_schema
, source_table.relname as source_table
, pg_attribute.attname as column_name
FROM pg_depend
JOIN pg_rewrite ON pg_depend.objid = pg_rewrite.oid
JOIN pg_class as dependent_view ON pg_rewrite.ev_class = dependent_view.oid
JOIN pg_class as source_table ON pg_depend.refobjid = source_table.oid
JOIN pg_attribute ON pg_depend.refobjid = pg_attribute.attrelid
    AND pg_depend.refobjsubid = pg_attribute.attnum
JOIN pg_namespace dependent_ns ON dependent_ns.oid = dependent_view.relnamespace
JOIN pg_namespace source_ns ON source_ns.oid = source_table.relnamespace
WHERE
source_ns.nspname = 'my_schema'
AND source_table.relname = 'my_table'
AND pg_attribute.attnum > 0
ORDER BY 1,2
LIMIT 10;

+------------------+----------------+---------------+--------------+-------------+
| dependent_schema | dependent_view | source_schema | source_table | column_name |
+------------------+----------------+---------------+--------------+-------------+
```

使用 IF EXISTS 丟棄資料表

下列範例會捨棄 FEEDBACK 資料表 (如存在的話)，或不執行任何動作，並於資料表不存在時傳回訊息：

```
drop table if exists feedback;
```

DROP USER

從資料庫捨棄使用者。使用單一 DROP USER 命令即可捨棄多個使用者。您必須是資料庫超級使用者，或具有 DROP USER 權限才能執行此命令。

語法

```
DROP USER [ IF EXISTS ] name [, ... ]
```

參數

IF EXISTS

此子句會指出，若指定的使用者不存在，則命令不應進行任何變更，且應傳回使用者不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在編寫指令碼時很實用，如此指令碼就不會因為 DROP USER 對不存在的使用者執行而失敗。

name

要移除的使用者名稱。您可以指定多個使用者，並以逗號分隔每個使用者名稱。

使用須知

您不能捨棄名為 rdsdb 的使用者或通常命名為 awsuser 或 admin 的資料庫管理員使用者。

若使用者擁有任何資料庫物件，像是結構描述、資料庫、資料表或檢視，或是使用者具有資料庫、資料表、資料欄或群組的任何權限，則無法捨棄使用者。如果您嘗試捨棄這類使用者，則會收到下列錯誤之一。

```
ERROR: user "username" can't be dropped because the user owns some object [SQL
 State=55006]

ERROR: user "username" can't be dropped because the user has a privilege on some object
 [SQL State=55006]
```

如需如何尋找資料庫使用者擁有之物件的詳細指示，請參閱知識中心內的如何解決 Amazon Redshift 中的「使用者無法捨棄」錯誤？。

> **Note**
>
> Amazon Redshift 在捨棄使用者之前，只會檢查目前資料庫。若使用者擁有資料庫物件，或具有其他資料庫中物件的權限，DROP USER 則不會傳回錯誤。若您捨棄的使用者擁有其他資料庫中的物件，則這些物件的擁有者會變更為「不明」。

若使用者擁有物件，請先捨棄物件或將其所有權變更為其他使用者，再捨棄原始使用者。如果使用者具有物件的權限，請先撤銷權限，再捨棄使用者。以下範例顯示在捨棄使用者前，先捨棄物件、變更所有權，以及撤銷權限。

```
drop database dwdatabase;
alter schema dw owner to dwadmin;
revoke all on table dwtable from dwuser;
drop user dwuser;
```

範例

以下範例會捨棄名為 paulo 的使用者：

```
drop user paulo;
```

以下範例會捨棄兩個使用者，paulo 和 martha：

```
drop user paulo, martha;
```

以下範例會捨棄使用者 paulo (如存在的話)，或不執行任何動作，並於該使用者帳戶不存在時傳回訊息：

```
drop user if exists paulo;
```

DROP VIEW

從資料庫移除檢視。使用單一 DROP VIEW 命令即可捨棄多個檢視。此命令無法還原。

所需權限

以下是 DROP VIEW 所需的權限：

- 超級使用者
- 具有 DROP VIEW 權限的使用者
- 檢視擁有者

語法

```
DROP VIEW [ IF EXISTS ] name [, ... ] [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

此子句會指出，若指定的檢視不存在，則命令不應進行任何變更，且應傳回檢視不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在編寫指令碼時很實用，如此指令碼就不會因為 DROP VIEW 對不存在的檢視執行而失敗。

name

要移除的檢視名稱。

CASCADE

此子句指出，自動捨棄取決於檢視的物件，例如其他檢視。

若要建立不相依於其他資料庫物件的檢視，例如檢視和資料表，請在檢視定義中包含 WITH NO SCHEMA BINDING 子句。如需詳細資訊，請參閱 CREATE VIEW。

請注意，如果您包含 CASCADE，而刪除的資料庫物件計數為十個或更多執行，您的資料庫用戶端可能不會在摘要結果中列出所有刪除的物件。這通常是因為 SQL 用戶端工具對傳回的結果有預設限制。

RESTRICT

此子句指出，若有任何物件相依於檢視，則不捨棄檢視。這是預設動作。

範例

下列範例會捨棄名為 event 的檢視：

```
drop view event;
```

若要移除有相依性的檢視，請使用 CASCADE 選項。例如，假設我們有名為 EVENT 的資料表。然後使用 CREATE VIEW 命令建立 EVENT 資料表的 eventview 檢視，如下列範例所示：

```
create view eventview as
select dateid, eventname, catid
from event where catid = 1;
```

現在我們建立另一個檢視，名為 myeventview，這是根據第一個檢視 eventview 所建立：

```
create view myeventview as
select eventname, catid
from eventview where eventname <> ' ';
```

此時已建立兩個檢視：eventview 和 myeventview。

myeventview 檢視是子檢視，eventview 是其父檢視。

若要刪除 eventview 檢視，明顯可使用下列命令：

```
drop view eventview;
```

請注意，如果您在此情況下執行此命令，將會收到下列錯誤：

```
drop view eventview;
ERROR: can't drop view eventview because other objects depend on it
HINT: Use DROP ... CASCADE to drop the dependent objects too.
```

若要解決此情況，請執行下列命令 (如錯誤訊息中所建議)：

```
drop view eventview cascade;
```

eventview 和 myeventview 現在都已成功捨棄。

下列範例會捨棄 eventview 檢視 (如存在的話)，或不執行任何動作，並於檢視不存在時傳回訊息：

```
drop view if exists eventview;
```

結束

遞交目前交易。與 COMMIT 命令執行完全相同的功能。

如需詳細文件，請參閱 COMMIT。

語法

```
END [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。

TRANSACTION

選用的關鍵字；WORK 和 TRANSACTION 為同義詞。

範例

以下範例全都會結束交易區塊並遞交交易：

```
end;
```

```
end work;
```

```
end transaction;
```

在這些命令之後，Amazon Redshift 會結束交易區塊並遞交變更。

EXECUTE

執行先前預備的陳述式。

語法

```
EXECUTE plan_name [ (parameter [, ...]) ]
```

參數

plan_name

要執行的預備陳述式名稱。

parameter

預備陳述式之參數的實際值。這個表達式所產生值的類型必須與建立預備陳述式的 PREPARE 命令中，為此參數位置所指定的資料類型相容。

使用須知

EXECUTE 是用來執行先前的預備陳述式。由於預備陳述式僅於工作階段期間存在，因此預備陳述式必須已由在目前工作階段之前執行的 PREPARE 陳述式建立。

如果先前的 PREPARE 陳述式指定了一些參數，則必須將一組相容的參數傳遞至 EXECUTE 陳述式，否則 Amazon Redshift 會傳回錯誤。與函數不同的是，預備陳述式不會根據指定的參數類型或數目過載；預備陳述式的名稱必須在資料庫工作階段內是唯一的。

對預備陳述式發出 EXECUTE 命令時，Amazon Redshift 可能會選擇先修改查詢執行計畫 (依據指定的參數值改善效能)，再執行預備陳述式。此外，每次重頭執行預備陳述式時，Amazon Redshift 都會根據隨 EXECUTE 陳述式指定的不同參數值，再次修改查詢執行計畫。若要檢查 Amazon Redshift 為任何特定 EXECUTE 陳述式選擇的查詢執行計畫，請使用 EXPLAIN 命令。

如需建立及使用預備陳述式的範例和詳細資訊，請參閱 PREPARE。

另請參閱

DEALLOCATE, PREPARE

EXPLAIN

顯示查詢陳述式的執行計畫，但不執行查詢。如需有關查詢分析工作流程的資訊，請參閱 查詢分析工作流程。

語法

```
EXPLAIN [ VERBOSE ] query
```

參數

詳細

顯示完整查詢計畫，不只是摘要而已。

query

要說明的查詢陳述式。查詢可以是 SELECT、INSERT、CREATE TABLE AS、UPDATE 或 DELETE 陳述式。

使用須知

EXPLAIN 效能有時會受到建立臨時資料表所需的時間影響。例如，使用通用子表達式最佳化的查詢需建立並分析臨時資料表，以便傳回 EXPLAIN 輸出。查詢計畫取決於臨時資料表的結構描述和統計資訊。因此，此類型查詢的 EXPLAIN 命令可能需要比預期更長的執行時間。

您只能針對下列命令使用 EXPLAIN：

- SELECT
- SELECT INTO
- CREATE TABLE AS
- INSERT
- UPDATE
- DELETE

若您將 EXPLAIN 命令用於其他 SQL 命令，例如資料定義語言 (DDL) 或資料庫操作，則此命令將會失敗。

Amazon Redshift 會使用 EXPLAIN 輸出的相對單位成本來選擇查詢計畫。Amazon Redshift 會比較各種資源估算值的大小來決定計畫。

查詢計劃和執行步驟

特定 Amazon Redshift 查詢陳述式的執行計畫會將查詢的執行和計算細分成一系列分散的步驟和資料表操作，這些最後會產生查詢的最終結果集。如需查詢計劃的資訊，請參閱 查詢處理。

下表摘要說明 Amazon Redshift 在開發使用者提交執行的任何查詢執行計畫時可使用的步驟。

EXPLAIN 運算子	查詢執行步驟	描述
SCAN：		
Sequential Scan	scan	Amazon Redshift 關聯式掃描或資料庫掃描運算子或步驟。從開頭到結尾循序掃描整個資料表；如 WHERE 子句中有指定，也會評估每個資料列的查詢限制條件 (篩選條件)。還會用來執行 INSERT、UPDATE 和 DELETE 陳述式。
JOINS：Amazon Redshift 會根據要聯結資料表的實體設計、聯結所需資料的位置，以及查詢本身的特定屬性來使用不同的聯結運算子。子查詢掃描 – 子查詢掃描和附加會用來執行 UNION 查詢。		
巢狀迴路	nloop	最差聯結；主要用於交叉聯結 (笛卡兒乘積；無聯結條件) 和一些不相等聯結中。
雜湊聯結	hjoin	同樣用於內部聯結和左右外部聯結，且通常會較巢狀迴路聯結更快速。雜湊聯結會讀取外部資料表、雜湊聯結資料欄，並在內部雜湊表中尋找相符項目。步驟可溢寫至磁碟。(hjoin 的內部輸出可以是磁碟型的雜湊步驟。)
合併聯結	mjoin	同樣用於內部聯結和外部聯結 (針對在聯結資料欄上進行分佈和排序的聯結資料表)。通常是速度最快的 Amazon Redshift 聯結演算法 (不納入其他成本考量的情況下)。
AGGREGATION：運算子和步驟，用於牽涉到彙整函數和 GROUP BY 操作的查詢。		
Aggregate	aggr	純量彙整函數的運算子/步驟。
HashAggregate	aggr	分組彙整函數的運算子/步驟。可從磁碟透過溢寫至磁碟的雜湊資料表操作。
GroupAggregate	aggr	有時會針對分組彙整查詢選擇的運算子，在 force_hash_grouping 設定的 Amazon Redshift 組態設定為關閉的情況下。
SORT：在查詢必須排序或合併結果集時使用的運算子和步驟。		

EXPLAIN 運算子	查詢執行步驟	描述
Sort	sort	Sort 會執行 ORDER BY 子句及其他操作所指定的排序，例如 UNION 和聯結。可從磁碟操作。
合併	merge	根據衍生自平行執行之操作的中繼排序結果，產生查詢的最終排序結果。
EXCEPT、INTERSECT 和 UNION 操作：		
SetOp 除了 [獨特]	hjoin	用於 EXCEPT 查詢。根據輸入雜湊可以是磁碟型的事實，可從磁碟操作。
Hash Intersect [Distinct]	hjoin	用於 INTERSECT 查詢。根據輸入雜湊可以是磁碟型的事實，可從磁碟操作。
Append [All |Distinct]	save	Append 搭配 Subquery Scan 實作 UNION 和 UNION ALL 查詢。可根據 "save" 從磁碟操作。
其他：		
Hash	hash	用於內部聯結和左右外部聯結 (對雜湊聯結提供輸入)。Hash 運算子會為聯結的內部資料表建立雜湊資料表 (內部資料表是在兩個資料表的聯結中，要查看其中是否有相符項目的資料表，通常是兩個資料表中較小的)。
限制	limit	評估 LIMIT 子句。
Materialize	save	具體化資料列以輸入至巢狀迴路聯結和一些合併聯結。可從磁碟操作。
--	parse	在載入時用來剖析文字輸入資料。
--	project	用來重新安排資料欄和表達式，也就是專案資料。
結果	--	執行未牽涉任何資料表存取的純量函數。
--	return	將資料列傳回至領導者或用戶端。

EXPLAIN 運算子	查詢執行步驟	描述
Subplan	--	用於特定子查詢。
唯一	unique	消除 SELECT DISTINCT 和 UNION 查詢中的重複項目。
視窗	window	運算彙整和排名視窗函數。可從磁碟操作。
網路操作：		
Network (Broadcast)	bcast	Broadcast 也是 Join Explain 運算子和步驟的屬性。
Network (Distribute)	dist	將資料列分佈至運算節點供資料倉儲叢集進行平行處理。
Network (Send to Leader)	return	將結果傳回領導者供進一步處理。
DML 操作 (修改資料的運算子)：		
Insert (using Result)	insert	插入資料。
Delete (Scan + Filter)	刪除	刪除資料。可從磁碟操作。
Update (Scan + Filter)	delete, insert	實作為 delete 和 Insert。

針對 RLS 使用 EXPLAIN

如果查詢包含受資料列層級安全性 (RLS) 原則限制的資料表，EXPLAIN 會顯示特殊的 RLS 節點。SecureScan Amazon Redshift 也會將相同的節點類型記錄到 STL_EXPLAIN 系統資料表中。EXPLAIN 不會展現適用於 dim_tbl 的 RLS 述詞。RLS SecureScan 節點類型可做為指標，表示執行計畫包含目前使用者無法看到的其他作業。

下列範例說明 RLS SecureScan 節點。

```
EXPLAIN
SELECT D.cint
FROM fact_tbl F INNER JOIN dim_tbl D ON F.k_dim = D.k
WHERE F.k_dim / 10 > 0;
```

```
                                QUERY PLAN
------------------------------------------------------------------------
 XN Hash Join DS_DIST_ALL_NONE  (cost=0.08..0.25 rows=1 width=4)
   Hash Cond: ("outer".k_dim = "inner"."k")
   ->  *XN* *RLS SecureScan f  (cost=0.00..0.14 rows=2 width=4)*
         Filter: ((k_dim / 10) > 0)
   ->  XN Hash  (cost=0.07..0.07 rows=2 width=8)
         ->  XN Seq Scan on dim_tbl d  (cost=0.00..0.07 rows=2 width=8)
               Filter: (("k" / 10) > 0)
```

為了對受 RLS 限制的查詢計畫進行全面調查，Amazon Redshift 提供了 EXPLAIN RLS 系統許可。已授予此權限的使用者可以檢查也包含 RLS 述詞的完整查詢計畫。

下列範例說明 RLS SecureScan 節點下方的其他「序號掃描」也包含 RLS 原則述詞 (k_dim > 1)。

```
EXPLAIN SELECT D.cint
FROM fact_tbl F INNER JOIN dim_tbl D ON F.k_dim = D.k
WHERE F.k_dim / 10 > 0;
                                  QUERY PLAN
------------------------------------------------------------------------------
 XN Hash Join DS_DIST_ALL_NONE  (cost=0.08..0.25 rows=1 width=4)
   Hash Cond: ("outer".k_dim = "inner"."k")
   *->  XN RLS SecureScan f  (cost=0.00..0.14 rows=2 width=4)
         Filter: ((k_dim / 10) > 0)*
         ->  *XN* *Seq Scan on fact_tbl rls_table  (cost=0.00..0.06 rows=5 width=8)
               Filter: (k_dim > 1)*
   ->  XN Hash  (cost=0.07..0.07 rows=2 width=8)
         ->  XN Seq Scan on dim_tbl d  (cost=0.00..0.07 rows=2 width=8)
               Filter: (("k" / 10) > 0)
```

雖然授予使用者 EXPLAIN RLS 許可，但 Amazon Redshift 也會在 STL_EXPLAIN 系統資料表中記錄包含 RLS 述詞的完整查詢計畫。若在未授予此許可的情況下執行查詢，查詢將在沒有 RLS 內部元件的情況下記錄。授予或移除 EXPLAIN RLS 許可不會改變 Amazon Redshift 針對先前查詢記錄到 STL_EXPLAIN 的內容。

AWS Lake Formation-RLS 保護的 Redshift 關係

下列範例說明 LF SecureScan 節點，您可以使用此節點來檢視 Lake 格式-RLS 關係。

```
EXPLAIN
SELECT *
FROM lf_db.public.t_share
WHERE a > 1;
```

```
QUERY PLAN
------------------------------------------------------------
XN LF SecureScan t_share  (cost=0.00..0.02 rows=2 width=11)
(2 rows)
```

範例

> **Note**
>
> 在這些範例中，範例輸出可能因 Amazon Redshift 組態而有所不同。

以下範例會針對從 EVENT 和 VENUE 資料表選取 EVENTID、EVENTNAME、VENUEID 和 VENUENAME 的查詢傳回查詢計畫：

```
explain
select eventid, eventname, event.venueid, venuename
from event, venue
where event.venueid = venue.venueid;
```

```
                               QUERY PLAN
------------------------------------------------------------------------
XN Hash Join DS_DIST_OUTER  (cost=2.52..58653620.93 rows=8712 width=43)
Hash Cond: ("outer".venueid = "inner".venueid)
->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=23)
->  XN Hash  (cost=2.02..2.02 rows=202 width=22)
->  XN Seq Scan on venue  (cost=0.00..2.02 rows=202 width=22)
(5 rows)
```

以下範例會傳回相同查詢的查詢計畫，但包含詳細輸出：

```
explain verbose
select eventid, eventname, event.venueid, venuename
from event, venue
where event.venueid = venue.venueid;
```

```
                               QUERY PLAN
------------------------------------------------------------------------
{HASHJOIN
:startup_cost 2.52
```

```
:total_cost 58653620.93
:plan_rows 8712
:plan_width 43
:best_pathkeys <>
:dist_info DS_DIST_OUTER
:dist_info.dist_keys (
TARGETENTRY
{
VAR
:varno 2
:varattno 1
...

XN Hash Join DS_DIST_OUTER  (cost=2.52..58653620.93 rows=8712 width=43)
Hash Cond: ("outer".venueid = "inner".venueid)
->  XN Seq Scan on event  (cost=0.00..87.98 rows=8798 width=23)
->  XN Hash  (cost=2.02..2.02 rows=202 width=22)
->  XN Seq Scan on venue  (cost=0.00..2.02 rows=202 width=22)
(519 rows)
```

以下範例會傳回 CREATE TABLE AS (CTAS) 陳述式的查詢計畫：

```
explain create table venue_nonulls as
select * from venue
where venueseats is not null;

QUERY PLAN
-------------------------------------------------------------
XN Seq Scan on venue  (cost=0.00..2.02 rows=187 width=45)
Filter: (venueseats IS NOT NULL)
(2 rows)
```

FETCH

使用資料指標擷取資料列。如需宣告資料指標的相關資訊，請參閱 DECLARE。

FETCH 會根據資料指標內目前的位置擷取資料列。建立資料指標時，它會定位在第一列前面。FETCH 之後，資料指標會定位在擷取的最後一列。如果 FETCH 執行超出可用資料列的結尾，例如接在 FETCH ALL 之後，資料指標會留在最後一列後面。

FORWARD 0 會擷取目前資料列，但不移動資料指標；也就是說，它會擷取最近擷取的資料列。如果資料指標定位在第一列前面或最後一列後面，則不會傳回任何資料列。

擷取資料指標的第一列時，整個結果集會在領導節點上、記憶體中或磁碟上具體化 (如有需要)。由於在大型結果集內使用資料指標可能會對效能造成負面影響，因此建議您盡量使用替代方式。如需詳細資訊，請參閱 使用游標時的效能考量。

如需詳細資訊，請參閱 DECLARE、CLOSE。

語法

```
FETCH [ NEXT | ALL | {FORWARD [ count | ALL ] } ] FROM cursor
```

參數

NEXT

擷取下一列。此為預設值。

ALL

擷取所有剩餘的資料列。(與 FORWARD ALL 相同。) 單一節點的叢集不支援 ALL。

FORWARD [count | ALL]

擷取下一個 count 資料列，或所有剩餘的資料列。FORWARD 0 會擷取目前資料列。若是單一節點叢集，count 的最大值為 1000。單一節點的叢集不支援 FORWARD ALL。

cursor

新資料指標的名稱。

FETCH 範例

以下範例會宣告名為 LOLLAPALOOZA 的資料指標來選取 Lollapalooza 活動的銷售資訊，然後使用資料指標從結果集擷取資料列：

```
-- Begin a transaction

begin;

-- Declare a cursor

declare lollapalooza cursor for
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
```

```
from sales, event
where sales.eventid = event.eventid
and eventname='Lollapalooza';

-- Fetch the first 5 rows in the cursor lollapalooza:

fetch forward 5 from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-05-01 19:00:00 |   92.00000000 |       3
 Lollapalooza | 2008-11-15 15:00:00 |  222.00000000 |       2
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       3
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       4
 Lollapalooza | 2008-04-17 15:00:00 |  239.00000000 |       1
(5 rows)

-- Fetch the next row:

fetch next from lollapalooza;

  eventname   |      starttime      | costperticket | qtysold
--------------+---------------------+---------------+---------
 Lollapalooza | 2008-10-06 14:00:00 |  114.00000000 |       2

-- Close the cursor and end the transaction:

close lollapalooza;
commit;
```

GRANT

定義使用者或角色的存取權限。

許可包括存取選項，例如能夠讀取資料表和檢視中的資料、寫入資料、建立資料表及捨棄資料表。使用此命令可提供資料表、資料庫、結構描述、函數、程序、語言或資料欄的特定許可。若要撤銷資料庫物件的許可，請使用 REVOKE 命令。

權限也包括下列資料共用生產者存取選項：

- 授予取用者命名空間和帳戶的資料共用存取權。
- 透過在資料共用中新增或移除物件，授予變更資料共用的權限。

- 透過從資料共用中新增或移除取用者命名空間，授予資料共用的權限。

資料共用取用者存取選項如下：

- 授予使用者對從資料共用建立的資料庫，或指向此類資料庫的外部結構描述的完全存取權。
- 授予使用者從資料共用建立之資料庫的物件層級權限，就像您對本機資料庫物件一樣。若要授予這個層級的權限，您必須在從資料共用建立資料庫時使用 WITH PERSOMES 子句。如需詳細資訊，請參閱 CREATE DATABASE。

如需資料共用權限的詳細資訊，請參閱 共用資料共用。

您也可以授予角色來管理資料庫許可，並控制使用者可以對您的資料執行哪些動作。透過定義角色並將角色指派給使用者，您可以限制這些使用者可執行的動作，例如限制使用者只能使用 CREATE TABLE 和 INSERT 命令。如需 CREATE ROLE 命令的相關資訊，請參閱 the section called “CREATE ROLE”。Amazon Redshift 具有一些系統定義的角色，您也可以使用這些角色將特定許可授予使用者。如需詳細資訊，請參閱 the section called “Amazon Redshift 系統定義角色”。

您只能對使用 ON SCHEMA 語法的資料庫使用者和使用者群組執行外部結構描述的 GRANT 或 REVOKE USAGE 許可。搭配使用 ON 外部結構描述時 AWS Lake Formation，您只能授與和撤銷 AWS Identity and Access Management (IAM) 角色的權限。如需許可的清單，請參閱語法。

若為預存程序，您唯一可授予的許可是 EXECUTE。

您無法在交易區塊 (BEGIN ... END) 內執行 GRANT (針對外部資源)。如需交易的相關資訊，請參閱 可序列化隔離。

若要查看使用者被授予的資料庫許可，請使用 HAS_DATABASE_PRIVILEGE。若要查看使用者被授予的結構描述許可，請使用 HAS_SCHEMA_PRIVILEGE。若要查看使用者被授予的資料表許可，請使用 HAS_TABLE_PRIVILEGE。

語法

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | ALTER | TRUNCATE }
 [,...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { { CREATE | TEMPORARY | TEMP | ALTER } [,...] | ALL [ PRIVILEGES ] }
```

```
    ON DATABASE db_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { { CREATE | USAGE | ALTER } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
 FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
 PROCEDURES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT USAGE
    ON LANGUAGE language_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

授予資料表的資料欄層級權限

下列語法適用於 Amazon Redshift 資料表和檢視上的資料欄層級許可。

```
GRANT { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
 ( column_name [,...] ) }
     ON { [ TABLE ] table_name [, ...] }

     TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

授予 ASSUMEROLE 許可

以下語法可將 ASSUMEROLE 許可授予具有指定角色的使用者和群組。若要開始使用 ASSUMEROLE 權限，請參閱 授予 ASSUMEROL 許可的使用須知。

```
GRANT ASSUMEROLE
       ON { 'iam_role' [, ...] | default | ALL }
```

```
    TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
    FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL } [, ...]
```

授予 Redshift Spectrum 與 Lake Formation 整合的許可

以下是 Redshift Spectrum 與 Lake Formation 整合的語法。

```
GRANT { SELECT | ALL [ PRIVILEGES ] } ( column_list )
    ON EXTERNAL TABLE schema_name.table_name
    TO { IAM_ROLE iam_role } [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | ALTER | DROP | DELETE | INSERT }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL TABLE schema_name.table_name [, ...]
    TO { { IAM_ROLE iam_role } [, ...] | PUBLIC } [ WITH GRANT OPTION ]

GRANT { { CREATE | ALTER | DROP }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL SCHEMA schema_name [, ...]
    TO { IAM_ROLE iam_role } [, ...] [ WITH GRANT OPTION ]
```

授予資料共用許可

生產者端資料共用權限

以下是使用 GRANT 將 ALTER 或 SHARE 權限授予使用者或角色的語法。使用者可以使用 ALTER 權限變更資料共用，或者使用 SHARE 權限將使用授予取用者。ALTER 和 SHARE 是您可以授予使用者和使用者群組的唯一權限。

```
GRANT { ALTER | SHARE } ON DATASHARE datashare_name
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

以下是在 Amazon Redshift 上使用 GRANT 授予資料共用使用許可的語法 您可以使用 USAGE 許可將資料共用的存取權授予取用者。您無法將此許可授予使用者或使用者群組。此許可也不支援對 GRANT 陳述式使用 WITH GRANT OPTION。只有先前因為資料共用而被授予 SHARE 許可的使用者或使用者群組才能執行此類型的 GRANT 陳述式。

```
GRANT USAGE
    ON DATASHARE datashare_name
    TO NAMESPACE 'namespaceGUID' | ACCOUNT 'accountnumber' [ VIA DATA CATALOG ]
```

以下是如何將資料共用使用權授予 Lake Formation 帳戶的範例。

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012' VIA DATA CATALOG;
```

取用者端資料共用權限

以下是 GRANT 資料共用使用許可的語法，用於特定資料庫或從資料共用建立的結構描述。

取用者存取從資料共用建立資料庫所需的進一步權限，取決於是否使用 WITH PERMISSIONS 子句從資料共用建立資料庫的 CREATE DATABASE 命令。如需有關 CREATE DATABASE 命令和 WITH PERMISSIONS 子句的詳細資訊，請參閱 CREATE DATABASE。

不使用 WITH PERMISSIONS 子句建立的資料庫

當您在沒有 WITH PERMISSIONS 子句之從資料共用建立的資料庫上授予 USAGE 時，您不需要對共用資料庫中的物件個別授予權限。在沒有 WITH PERMISSIONS 子句之從資料共用建立的資料庫上授予使用權的實體，會自動存取資料庫中的所有物件。

使用 WITH PERMISSIONS 子句建立的資料庫

當您在資料庫上授予 USAGE 時，其中共用資料庫是使用 WITH PERMISSIONS 子句從資料共用建立的，取用者端身分仍然必須被授予共用資料庫中資料庫物件的相關權限才能存取該物件，就像您授予權限給本機資料庫物件一樣。若要將權限授予從資料共用建立之資料庫中的物件，請使用三部分語法 `database_name.schema_name.object_name`。若要將權限授予指向共用資料庫內共用結構描述的外部結構描述中的物件，請使用兩部分語法 `schema_name.object_name`。

```
GRANT USAGE ON { DATABASE shared_database_name [, ...] | SCHEMA shared_schema}
    TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

授予限定範圍權限

作用域權限可讓您將權限授與資料庫或結構描述中某個類型之所有物件的使用者或角色。具有範圍權限的使用者和角色對資料庫或結構描述中所有目前和 future 物件都具有指定的權限。

以下是將限定範圍權限授予使用者或角色的語法。如需有關範圍權限的詳細資訊，請參閱。限定範圍權限

```
GRANT { CREATE | USAGE | ALTER } [,...] | ALL [ PRIVILEGES ] }
FOR SCHEMAS IN
DATABASE db_name
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

GRANT
```

```
{ { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
 [, ...] } | ALL [PRIVILEGES] } }
FOR TABLES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name} [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
FOR FUNCTIONS IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name | } [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
FOR PROCEDURES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name | } [, ...]

GRANT USAGE
FOR LANGUAGES IN
{DATABASE db_name}
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]
```

請注意，作用域權限不會區分函數和程序的權限。例如，下列陳述式會授bob與結構描述中函數和程序的EXECUTE權限Sales_schema。

```
GRANT EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema TO bob;
```

授予機器學習許可

以下是 Amazon Redshift 上的機器學習模型許可語法。

```
GRANT CREATE MODEL
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON MODEL model_name [, ...]

    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

授予角色許可

以下是在 Amazon Redshift 上使用授予角色許可的語法。

```
GRANT { ROLE role_name } [, ...] TO { { user_name [ WITH ADMIN OPTION ] } |
 ROLE role_name }[, ...]
```

以下是在 Amazon Redshift 上對角色授予系統許可的語法。

```
GRANT
  {
    { CREATE USER | DROP USER | ALTER USER |
    CREATE SCHEMA | DROP SCHEMA |
    ALTER DEFAULT PRIVILEGES |
    ACCESS CATALOG |
    CREATE TABLE | DROP TABLE | ALTER TABLE |
    CREATE OR REPLACE FUNCTION | CREATE OR REPLACE EXTERNAL FUNCTION |
    DROP FUNCTION |
    CREATE OR REPLACE PROCEDURE | DROP PROCEDURE |
    CREATE OR REPLACE VIEW | DROP VIEW |
    CREATE MODEL | DROP MODEL |
    CREATE DATASHARE | ALTER DATASHARE | DROP DATASHARE |
    CREATE LIBRARY | DROP LIBRARY |
    CREATE ROLE | DROP ROLE |
    TRUNCATE TABLE
    VACUUM | ANALYZE | CANCEL }[, ...]
  }
  | { ALL [ PRIVILEGES ] }
TO { ROLE role_name } [, ...]
```

針對資料列層級安全性政策篩選授予解釋許可

以下語法可用來授予許可，以解釋 EXPLAIN 計畫中查詢的資料列層級安全性政策篩選。您可以使用 REVOKE 陳述式撤銷權限。

```
GRANT EXPLAIN RLS TO ROLE rolename
```

以下是授予許可以略過查詢資料列層級安全性政策的語法。

```
GRANT IGNORE RLS TO ROLE rolename
```

將 RLS 查閱資料表的許可授予政策物件

以下是對指定資料列層級安全性政策授予許可的語法。

```
GRANT SELECT ON [ TABLE ] table_name [, ...]
TO RLS POLICY policy_name [, ...]
```

參數

SELECT

授予許可以使用 SELECT 陳述式從資料表或檢視選取資料。另外也需要 SELECT 許可，才能參考 UPDATE 或 DELETE 操作的現有資料欄值。

INSERT

授予許可以使用 INSERT 陳述式或 COPY 陳述式將資料載入資料表中。

UPDATE

授予許可以使用 UPDATE 陳述式更新資料表資料欄。UPDATE 操作也需要 SELECT 許可，因為這些操作必須參考資料表資料欄，才能判斷要更新哪些資料列，或運算資料欄的新值。

DELETE

允許刪除資料表中的資料列。DELETE 操作也需要 SELECT 許可，因為這些操作必須參考資料表資料欄，才能判斷要刪除哪些資料列。

DROP

准許捨棄資料表。此權限適用於 Amazon Redshift 以及為 Lake Formation 啟用的。 AWS Glue Data Catalog

REFERENCES

授予建立外部索引鍵限制條件的許可。您需要在被動參考資料表與主動參考資料表上授予此許可；否則，使用者就無法建立限制條件。

ALTER

根據資料庫物件而定，將下列許可授予使用者或使用者群組：

- 對於資料表，ALTER 會授予修改資料表或檢視的許可。如需詳細資訊，請參閱 ALTER TABLE。
- 對於資料庫，ALTER 會授予修改資料庫的許可。如需詳細資訊，請參閱 ALTER DATABASE。
- 對於結構描述，ALTER 會授予修改結構描述的許可。如需詳細資訊，請參閱 ALTER SCHEMA。
- 對於外部資料表，ALTER 會授與變更已啟用 Lake Formation AWS Glue Data Catalog 的表格的權限。此許可僅適用於使用 Lake Formation 時。

TRUNCATE

允許截斷資料表 如果沒有此許可，只有資料表的擁有者或超級使用者可以截斷資料表。如需 TRUNCATE 命令的相關資訊，請參閱 the section called “TRUNCATE”。

ALL [PRIVILEGES]

對指定使用者或使用者群組一次授予所有可用許可。PRIVILEGES 關鍵字為選用。

GRANT ALL ON SCHEMA 不會授予外部結構描述的 CREATE 許可。

您可以將 ALL 權限授予對已啟用 Lake Formation 的表格。 AWS Glue Data Catalog 在這種情況下，系統會將 SELECT 和 ALTER 等個別許可記錄在 Data Catalog. 中。

ASSUMEROLE

對使用者、角色或具有指定角色的群組授予執行 COPY、UNLOAD、EXTERNAL FUNCTION 和 CREATE MODEL 命令的許可。執行指定的命令時，使用者、角色或群組會擔任該角色。若要開始使用 ASSUMEROLE 許可，請參閱 授予 ASSUMEROL 許可的使用須知。

ON [TABLE] table_name

授予資料表或檢視的指定許可。TABLE 關鍵字為選用。您可在一個陳述式中列出多個資料表和檢視。

ON ALL TABLES IN SCHEMA schema_name

授予所參考結構描述中所有資料表和檢視的指定許可。

(column_name [,...]) ON TABLE table_name

將 Amazon Redshift 資料表或檢視中指定資料欄的指定許可授予使用者、群組或 PUBLIC。

(column_list) ON EXTERNAL TABLE schema_name.table_name

為所參考結構描述中 Lake Formation 資料表之指定欄上的 IAM 角色授予指定許可。

ON EXTERNAL TABLE schema_name.table_name

為所參考結構描述中 Lake Formation 資料表上的 IAM 角色授予指定許可。

ON EXTERNAL SCHEMA schema_name

為所參考結構描述上的 IAM 角色授予指定許可。

ON iam_role

將指定許可授予 IAM 角色。

TO username

指出接收許可的使用者。

TO IAM_ROLE iam_role

指出接收許可的 IAM 角色。

WITH GRANT OPTION

指出接收許可的使用者可以接著將相同的許可授予他人。無法將 WITH GRANT OPTION 授予群組或 PUBLIC。

ROLE role_name

將許可授予角色。

GROUP group_name

將許可授予使用者群組。可以使用逗號分隔清單來指定多個使用者群組。

PUBLIC

對所有使用者授予指定的許可，包括之後建立的使用者。PUBLIC 代表永遠包含所有使用者的群組。個別使用者的許可是由授予 PUBLIC 的許可、授予使用者所屬之任何群組的許可，以及個別授予使用者的任何許可，三者加總所組成。

將 PUBLIC 授予 Lake Formation EXTERNAL TABLE 會導致向 Lake Formation 的每個人群組授予許可。

CREATE

根據資料庫物件而定，將下列許可授予使用者或使用者群組：

- 若是資料庫，CREATE 可讓使用者在資料庫內建立結構描述。
- 若是結構描述，CREATE 可讓使用者在結構描述內建立物件。若要重新命名物件，使用者必須具有 CREATE 許可並擁有要重新命名的物件。
- Amazon Redshift Spectrum 外部結構描述不支援 CREATE ON SCHEMA。若要授予外部結構描述中外部資料表的使用權，請將 USAGE ON SCHEMA 授予需要存取權的使用者。只有外部結構描述的擁有者或超級使用者才可在外部結構描述中建立外部資料表。若要轉移外部結構描述的所有權，請使用 ALTER SCHEMA 來變更擁有者。

TEMPORARY | TEMP

授予許可以在指定的資料庫中建立臨時資料表。若要執行 Amazon Redshift Spectrum 查詢，資料庫使用者必須具有在資料庫中建立臨時資料表的許可。

> **Note**
>
> 根據預設，建立臨時資料表的許可是依使用者在 PUBLIC 群組中的自動成員資格授予。若要移除任何使用者建立暫存資料表的許可，請撤銷 PUBLIC 群組的 TEMP 許可。然後將建立暫存資料表的許可明確授予特定使用者或使用者群組。

ON DATABASE db_name

授予資料庫的指定許可。

USAGE

授予特定結構描述的 USAGE 許可，即可讓使用者存取該結構描述中的物件。若是本機 Amazon Redshift 結構描述，對這些物件的特定動作必須另行授予 (例如資料表的 SELECT 或 UPDATE 許可)。根據預設，所有使用者對 PUBLIC 結構描述都具有 CREATE 和 USAGE 許可。

當您使用 ON SCHEMA 語法將 USAGE 授予外部結構描述時，您不需要對外部結構描述中的物件個別授予動作。對應的目錄許可會控制外部結構描述物件的精細許可。

ON SCHEMA schema_name

授予結構描述的指定許可。

Amazon Redshift Spectrum 外部結構描述不支援 GRANT ALL ON SCHEMA 中的 GRANT CREATE ON SCHEMA 和 CREATE 許可。若要授予外部結構描述中外部資料表的使用權，請將 USAGE ON SCHEMA 授予需要存取權的使用者。只有外部結構描述的擁有者或超級使用者才可在外部結構描述中建立外部資料表。若要轉移外部結構描述的所有權，請使用 ALTER SCHEMA 來變更擁有者。

EXECUTE ON ALL FUNCTIONS IN SCHEMA schema_name

授予所參考結構描述中所有函數的指定許可。

Amazon Redshift 不支援對 pg_catalog 命名空間中定義的 pg_proc 內建項目使用 GRANT 或 REVOKE 陳述式。

EXECUTE ON PROCEDURE procedure_name

授予特定預存程序的 EXECUTE 許可。由於預存程序名稱可以過載，因此您必須包含程序的引數清單。如需詳細資訊，請參閱 命名預存程序。

EXECUTE ON ALL PROCEDURES IN SCHEMA schema_name

授予所參考結構描述中所有預存程序的指定許可。

USAGE ON LANGUAGE language_name

授予某種語言的 USAGE 許可。

需具有 USAGE ON LANGUAGE 許可，才能透過執行 CREATE FUNCTION 命令建立使用者定義的函數 (UDF)。如需詳細資訊，請參閱 UDF 安全與權限。

需具有 USAGE ON LANGUAGE 許可，才能透過執行 CREATE PROCEDURE 命令建立預存程序。如需詳細資訊，請參閱 預存程序的安全和權限 。

若是 Python UDF，請使用 `plpythonu`。若是 SQL UDF，請使用 `sql`。若為預存程序，請使用 `plpgsql`。

FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL } [, ...]

指定 SQL 命令的授予權限。您可以指定 ALL 來授予 COPY、UNLOAD、EXTERNAL FUNCTION 和 CREATE MODEL 陳述式的許可。此子句僅適用於授予 ASSUMEROLE 許可。

ALTER

授予 ALTER 許可給使用者，使他們可以在資料共用中新增或移除物件，或設定屬性 PUBLICACCESSIBLE。如需詳細資訊，請參閱 ALTER DATASHARE。

SHARE

授予使用者和使用者群組將資料取用者新增至資料共用的許可。需要此許可才能讓特定取用者 (帳戶或命名空間) 從其叢集存取資料共用。消費者可以是相同或不同的 AWS 帳戶，具有與全域唯一識別碼 (GUID) 所指定的相同或不同叢集命名空間。

ON DATASHARE datashare_name

授予參考資料共用的指定許可。如需有關取用者存取控制精細度的資訊，請參閱 在 Amazon Redshift 中共用不同層級的資料。

USAGE

將 USAGE 授予相同帳戶內的取用者帳戶或命名空間時，帳戶內的特定取用者帳戶或命名空間可以以唯讀方式存取資料共用和資料共用的物件。

TO NAMESPACE 'clusternamespace GUID'

表示相同帳戶中的命名空間，取用者可以在其中接收資料共用的指定許可。命名空間使用 128 位元英數字元 GUID。

TO ACCOUNT 'accountnumber' [VIA DATA CATALOG]

表示另一個帳戶的編號，其中的取用者可以接收資料共用的指定許可。指定 'VIA DATA CATALOG' 表示您正在授予資料共用的使用權給 Lake Formation 帳戶。省略此參數表示您將使用權授予擁有叢集的帳戶。

ON DATABASE shared_database_name> [, ...]

在指定資料共用中建立的指定資料庫上授予指定使用許可。

ON SCHEMA shared_schema

在指定資料共用中建立的指定結構描述上授予指定許可。

FOR { SCHEMAS | TABLES | FUNCTIONS | PROCEDURES | LANGUAGES } IN

指定要被授予權限的資料庫物件。IN 後面的參數定義授予權限的範圍。

建立模型

將 CREATE MODEL 許可授予特定使用者或使用者群組。

ON MODEL model_name

授予特定模型的 EXECUTE 許可。

ACCESS CATALOG

授予檢視角色可存取之物件之相關中繼資料的權限。

{ role } [, ...]

要授予其他角色、使用者或 PUBLIC 的角色。

PUBLIC 代表永遠包含所有使用者的群組。個別使用者的許可是由授予 PUBLIC 的許可、授予使用者所屬之任何群組的許可，以及個別授予使用者的任何許可，三者加總所組成。

TO { { user_name [WITH ADMIN OPTION] } | role }[, ...]

將指定的角色授予具有 WITH ADMIN OPTION、其他角色或 PUBLIC 的指定使用者。

WITH ADMIN OPTION 子句會為所有承授者提供所有授予角色的管理選項。

EXPLAIN RLS TO ROLE rolename

授予許可，對角色解釋 EXPLAIN 計畫中查詢的資料列層級安全性政策篩選。

IGNORE RLS TO ROLE rolename

授予許可，以略過對角色查詢資料列層級安全性政策。

使用須知

若要進一步了解 GRANT 使用須知，請參閱 the section called “使用須知”。

範例

如需使用 GRANT 的範例，請參閱 the section called “範例”。

使用須知

若要授予物件的權限，您必須符合下列條件之一：

- 身為物件擁有者。
- 身為超級使用者。
- 具有該物件和權限的授予權限。

例如，以下命令可讓使用者 HR 在 employees 資料表上執行 SELECT 命令，並對其他使用者授予和撤銷相同的權限。

```
grant select on table employees to HR with grant option;
```

HR 無法授予 SELECT 以外任何操作的權限，也無法授予 employees 資料表以外任何資料表的權限。

在另一個範例中，以下命令可讓使用者 HR 在 employees 資料表上執行 ALTER 命令，並對其他使用者授予和撤銷相同的權限。

```
grant ALTER on table employees to HR with grant option;
```

HR 無法授予 ALTER 以外任何操作的權限，也無法授予 employees 資料表以外任何資料表的權限。

在檢視上授予權限並不表示具有基礎資料表的權限。同樣地，在結構描述上授予權限並不表示具有結構描述中資料表的權限。請改為明確授與基礎資料表的存取權限。

若要授與資料 AWS Lake Formation 表的權限，與資料表外部結構描述相關聯的 IAM 角色必須具有將權限授與外部資料表的權限。下列範例所建立的外部結構描述內含相關聯的 myGrantor IAM 角色。

該 myGrantor IAM 角色具備將許可授予他人的許可。GRANT 命令會使用與外部結構描述相關聯的 myGrantor IAM 角色許可，將許可授予 myGrantee IAM 角色。

```
create external schema mySchema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myGrantor'
create external database if not exists;
```

```
grant select
on external table mySchema.mytable
to iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

如果對 IAM 角色將授予所有權限 (GRANT ALL)，則啟用 Lake Formation 的相關 Data Catalog 中將授予個別權限。例如，執行下列 GRANT ALL 的結果是 Lake Formation 主控台中會顯示授予的個別權限 (SELECT、ALTER、DROP、DELETE 和 INSERT)。

```
grant all
on external table mySchema.mytable
to iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

無論是使用 GRANT 或 REVOKE 命令設定物件權限，超級使用者都能存取所有物件。

欄位層級存取控制的使用須知

下列使用須知適用於 Amazon Redshift 資料表和檢視上的資料欄層級權限。這些須知會描述資料表；除非我們明確提到例外，否則相同的須知會適用於檢視。

- 對於 Amazon Redshift 資料表，您只能在資料欄層級授予 SELECT 和 UPDATE 權限。對於 Amazon Redshift 檢視，您只能在資料欄層級授予 SELECT 權限。
- 在資料表上的資料欄層級 GRANT 內容中使用時，ALL 關鍵字是 SELECT 和 UPDATE 權限組合的同義詞。
- 如果您沒有資料表中所有資料欄的 SELECT 權限，執行 SELECT * 操作只會傳回您有權存取的資料欄。使用檢視時，SELECT * 作業會嘗試存取檢視中的所有欄。如果您沒有存取所有資料行的權限，這些查詢會失敗，並顯示權限遭拒錯誤。
- 在以下情況下，SELECT * 不會擴展到僅可存取的資料欄：
  - 您無法使用 SELECT * 建立僅具有可存取資料欄的一般檢視。
  - 您無法使用 SELECT * 建立僅具有可存取資料欄的具體化視觀表。

- 如果您有資料表或檢視的 SELECT 或 UPDATE 權限並新增一個資料欄，則您對資料表或檢視及其所有資料欄仍具有相同的權限。
- 只有資料表的擁有者或超級使用者可以授與資料欄層級權限。
- 資料欄層級權限不支援 WITH GRANT OPTION 子句。
- 您無法同時在資料表層級和資料欄層級擁有相同的權限。例如，使用者 `data_scientist` 不能同時擁有資料表 `employee` 和資料欄 `employee.department` 的 SELECT 權限。對資料表和資料表內的資料欄授與相同權限時，請考慮下列結果：
  - 如果使用者具有資料表的資料表層級權限，則在資料欄層級授與相同權限就沒有作用。
  - 如果使用者具有資料表的資料表層級權限，對資料表的一或多個資料欄撤銷相同的權限，則會傳回錯誤。請改為在資料表層級撤銷權限。
  - 如果使用者具有資料欄層級權限，在資料表層級授與相同的權限，則會傳回錯誤。
  - 如果使用者具有資料欄層級權限，在資料表層級撤銷相同的權限，則會撤銷資料表上所有資料欄的資料欄和資料表權限。
- 您無法授與延遲繫結檢視的資料欄層級權限。
- 若要建立具體化檢視，您必須具有基礎資料表的資料表層級 SELECT 權限。即使您擁有特定資料欄層級的權限，也無法針對這些資料欄建立具體化檢視。不過，您可以將 SELECT 權限授與具體化檢視的資料欄，與一般檢視類似。
- 若要查閱資料欄層級權限的授與，請使用 PG_ATTRIBUTE_INFO 檢視。

授予 ASSUMEROL 許可的使用須知

下列是在 Amazon Redshift 中授予 ASSUMEROLE 許可的使用須知。

您可以使用 ASSUMEROLE 許可控制資料庫使用者、角色或群組在命令上的 IAM 角色存取許可，例如 COPY、UNLOAD、EXTERNAL FUNCTION 或 CREATE MODEL 等命令。將 ASSUMEROLE 許可授予 IAM 角色的使用者、角色或群組之後，使用者、角色或群組就可以在執行命令時擔任該角色。ASSUMEROLE 許可可讓您視需要授予適當命令的存取權。

只有資料庫超級使用者可以授予或撤銷使用者、角色和群組的 ASSUMEROLE 許可。超級使用者始終保有 ASSUMEROLE 許可。

若要為使用者、角色和群組啟用 ASSUMEROLE 許可的使用權，超級使用者會執行下列兩個動作：

- 在叢集上執行下列陳述式一次：

```
revoke assumerole on all from public for all;
```

- 將 ASSUMEROLE 許可授予使用者、角色和群組，以使用適當的命令。

授予 ASSUMEROLE 許可時，您可以在 ON 子句中指定角色鏈結。您可以使用逗號來分隔角色鏈中的角色，例如：Role1,Role2,Role3。如果在授予 ASSUMEROLE 權限時指定角色鏈結，您必須在執行 ASSUMEROLE 許可授予的操作時指定角色鏈結。執行 ASSUMEROLE 許可授予的操作時，您無法指定角色鏈中的個別角色。例如，如果使用者、角色或群組被授予角色鏈結 Role1,Role2,Role3，則您無法指定僅 Role1 執行操作。

如果使用者嘗試執行 COPY、UNLOAD、EXTERNAL FUNCTION 或 CREATE MODEL 操作，但尚未獲得 ASSUMEROLE 許可，則會出現類似下列內容的訊息。

```
ERROR:  User awsuser does not have ASSUMEROLE permission on IAM role
 "arn:aws:iam::123456789012:role/RoleA" for COPY
```

若要列出已透過 ASSUMEROLE 許可授予 IAM 角色和命令存取權的使用者，請參閱。HAS_ASSUMEROLE_PRIVILEGE若要列出已授予所指定使用者的 IAM 角色和命令許可，請參閱 PG_GET_IAM_ROLE_BY_USER。若要列出已授予您指定 IAM 角色存取權的使用者、角色和群組，請參閱 PG_GET_GRANTEE_BY_IAM_ROLE。

授予機器學習許可的使用須知

您無法直接授予或撤銷與 ML 函數相關的許可。ML 函數屬於 ML 模型，而且許可是透過模型控制的。相反地，您可以授予 ML 模型相關的許可。下列範例會示範如何授予許可給所有使用者，以便執行與模型 customer_churn 相關聯的 ML 函數。

```
GRANT EXECUTE ON MODEL customer_churn TO PUBLIC;
```

您也可以將所有許可授予 ML 模型 customer_churn 的使用者。

```
GRANT ALL on MODEL customer_churn TO ml_user;
```

如果結構描述中有 ML 函數，則授予 ML 函數相關的 EXECUTE 許可將會失敗，即使該 ML 函數已經具有透過 GRANT EXECUTE ON MODEL 取得的 EXECUTE 許可也是一樣。我們建議您在使用 CREATE MODEL 命令時，使用個別的結構描述，將 ML 函數本身保留在不同的結構描述中。下列範例示範如何執行此動作。

```
CREATE MODEL ml_schema.customer_churn
FROM customer_data
```

```
TARGET churn
FUNCTION ml_schema.customer_churn_prediction
IAM_ROLE default
SETTINGS (
 S3_BUCKET 'your-s3-bucket'
);
```

範例

以下範例會將 SALES 資料表的 SELECT 權限授予使用者 fred。

```
grant select on table sales to fred;
```

以下範例會將 QA_TICKIT 結構描述中所有資料表的 SELECT 權限授予使用者 fred。

```
grant select on all tables in schema qa_tickit to fred;
```

以下範例會將 QA_TICKIT 結構描述中的所有結構描述權限授予使用者群組 QA_USERS。結構描述權限為 CREATE 和 USAGE。USAGE 會授予使用者存取結構描述中物件的權限，但不會授予物件的 INSERT 或 SELECT 這類權限。分別授與每個物件的權限。

```
create group qa_users;
grant all on schema qa_tickit to group qa_users;
```

以下範例會將 QA_TICKIT 結構描述中 SALES 資料表的所有權限授予群組 QA_USERS 中的所有使用者。

```
grant all on table qa_tickit.sales to group qa_users;
```

以下範例會將 QA_TICKIT 結構描述中 SALES 資料表的所有權限授予群組 QA_USERS 和 RO_USERS 中的所有使用者。

```
grant all on table qa_tickit.sales to group qa_users, group ro_users;
```

以下範例會將 QA_TICKIT 結構描述中 SALES 資料表的 DROP 權限授予群組 QA_USERS 中的所有使用者。

```
grant drop on table qa_tickit.sales to group qa_users;>
```

以下一系列命令說明，結構描述的存取權不會授予結構描述中資料表的權限。

```
create user schema_user in group qa_users password 'Abcd1234';
create schema qa_tickit;
create table qa_tickit.test (col1 int);
grant all on schema qa_tickit to schema_user;

set session authorization schema_user;
select current_user;


current_user
--------------
schema_user
(1 row)


select count(*) from qa_tickit.test;


ERROR: permission denied for relation test [SQL State=42501]


set session authorization dw_user;
grant select on table qa_tickit.test to schema_user;
set session authorization schema_user;
select count(*) from qa_tickit.test;


count
-------
0
(1 row)
```

以下一系列命令說明，檢視的存取權不代表能夠存取其基礎資料表。雖然名為 VIEW_USER 的使用者具有 VIEW_DATE 的所有權限，但這位使用者仍無法從 DATE 資料表選取。

```
create user view_user password 'Abcd1234';
create view view_date as select * from date;
grant all on view_date to view_user;
set session authorization view_user;
select current_user;
```

```
current_user
--------------
view_user
(1 row)


select count(*) from view_date;


count
-------
365
(1 row)


select count(*) from date;


ERROR:  permission denied for relation date
```

下列範例會將 cust_profile 資料表中 cust_name 和 cust_phone 資料欄的 SELECT 權限授予使用者 user1。

```
grant select(cust_name, cust_phone) on cust_profile to user1;
```

下列範例會將 cust_name 和 cust_phone 資料欄的 SELECT 權限，以及 cust_profile 資料表中 cust_contact_preference 資料欄的 UPDATE 權限授予 sales_group 群組。

```
grant select(cust_name, cust_phone), update(cust_contact_preference) on cust_profile to
 group sales_group;
```

下列範例顯示 ALL 關鍵字的使用方式，以將資料表 cust_profile 中三個資料欄的 SELECT 和 UPDATE 權限授予 sales_admin 群組。

```
grant ALL(cust_name, cust_phone,cust_contact_preference) on cust_profile to group
 sales_admin;
```

以下範例會將 cust_profile_vw 檢視中 cust_name 資料欄的 SELECT 權限授予 user2 使用者。

```
grant select(cust_name) on cust_profile_vw to user2;
```

授予資料共用存取權的範例

以下是顯示 GRANT 資料共用使用許可的範例，用於特定資料庫或從資料共用建立的結構描述。

在下列範例中，生產者端管理員會授予指定之命名空間的 salesshare 資料共用上的 USAGE 權限。

```
GRANT USAGE ON DATASHARE salesshare TO NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

在下列範例中，取用者端管理員會在 sales_db 至 Bob 上授予的 USAGE 權限。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

在下列範例中，取用者端管理員會將 sales_schema 結構描述的 GRANT USAGE 權限授予 Analyst_role 角色。sales_schema 是指向 sales_db 的外部結構描述。

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

此時，Bob 和 Analyst_role 可以存取 sales_schema 和 sales_db 中的所有資料庫物件。

下列範例顯示授予共用資料庫中物件的其他物件層級權限。只有當用於建立共用資料庫之 CREATE DATABASE 使用 WITH PERMISSIONS 子句，這些額外的權限才是必要的。如果 CREATE DATABASE 命令沒有使用 WITH PERMISSIONS，授予共用資料庫上的 USAGE 權限即是授予對該資料庫中所有物件的完全存取權限。

```
GRANT SELECT ON sales_db.sales_schema.tickit_sales_redshift to Bob;
```

授予限定範圍權限的範例

下列範例會將 Sales_db 資料庫中所有目前和未來結構描述的使用權授予 Sales 角色。

```
GRANT USAGE FOR SCHEMAS IN DATABASE Sales_db TO ROLE Sales;
```

下列範例會將 Sales_db 資料庫中所有目前和未來資料表的 SELECT 權限授予使用者 alice，並提供 alice 權限以授予 Sales_db 中資料表的限定範圍權限給其他使用者。

```
GRANT SELECT FOR TABLES IN DATABASE Sales_db TO alice WITH GRANT OPTION;
```

下列範例會將 Sales_schema 結構描述中函數的 EXECUTE 權限授予使用者 bob。

```
GRANT EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema TO bob;
```

下列範例會將 ShareDb 資料庫的 ShareSchema 結構描述中所有資料表的所有權限授予 Sales 角色。當指定結構描述時，您可以使用兩部分格式 database.schema 指定結構描述的資料庫。

```
GRANT ALL FOR TABLES IN SCHEMA ShareDb.ShareSchema TO ROLE Sales;
```

以下範例與前面的範例是相同的。您可以使用 DATABASE 關鍵字來指定資料庫，而不是使用兩部分格式。

```
GRANT ALL FOR TABLES IN SCHEMA ShareSchema DATABASE ShareDb TO ROLE Sales;
```

授予 ASSUMEROLE 權限的範例

以下是授予 ASSUMEROLE 權限的範例

下列範例顯示超級使用者在叢集上執行一次的 REVOKE 陳述式，以啟用使用者和群組的 ASSUMEROLE 權限。然後，超級使用者會將 ASSUMEROLE 權限授予使用者和群組，以取得適當的命令。有關如何為使用者和群組啟用 ASSUMEROLE 權限的資訊，請參閱 授予 ASSUMEROL 許可的使用須知。

```
revoke assumerole on all from public for all;
```

下列範例會將 ASSUMEROLE 權限授予使用者 reg_user1，讓 IAM 角色 Redshift-S3-Read 執行 COPY 操作。

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-S3-Read'
to reg_user1 for copy;
```

下列範例會將 ASSUMEROLE 權限授予使用者 reg_user1，讓 IAM 角色鏈 RoleB、RoleA 執行 UNLOAD 操作。

```
grant assumerole
on 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB'
to reg_user1
for unload;
```

以下是使用 IAM 角色鏈結 RoleA、RoleB 的 UNLOAD 命令範例。

```
unload ('select * from venue limit 10')
to 's3://companyb/redshift/venue_pipe_'
iam_role 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB';
```

下列範例會將 ASSUMEROLE 權限授予使用者 reg_user1，讓 IAM 角色 Redshift-Exfunc 建立外部函數。

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-Exfunc'
to reg_user1 for external function;
```

下列範例會將 ASSUMEROLE 權限授予使用者 reg_user1，讓 IAM 角色 Redshift-model 建立機器學習模型。

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-ML'
to reg_user1 for create model;
```

授予 ROLE 權限的範例

下列範例會將 sample_role1 授予 user1。

```
CREATE ROLE sample_role1;
GRANT ROLE sample_role1 TO user1;
```

下列範例會使用 WITH ADMIN OPTION 選項將 sample_role1 授予 user1、為 user1 設定目前工作階段，以及 user1 會將 sample_role1 授予 user2。

```
GRANT ROLE sample_role1 TO user1 WITH ADMIN OPTION;
SET SESSION AUTHORIZATION user1;
GRANT ROLE sample_role1 TO user2;
```

下列範例會將 sample_role1 授予 sample_role2。

```
GRANT ROLE sample_role1 TO ROLE sample_role2;
```

下列範例會將 sample_role2 授予 sample_role3 和 sample_role4。然後嘗試將 sample_role3 授予 sample_role1。

```
GRANT ROLE sample_role2 TO ROLE sample_role3;
GRANT ROLE sample_role3 TO ROLE sample_role2;
ERROR: cannot grant this role, a circular dependency was detected between these roles
```

下列範例會將 CREATE USER 系統權限授予 sample_role1。

```
GRANT CREATE USER TO ROLE sample_role1;
```

下列範例會將系統定義角色 sys:dba 授予 user1。

```
GRANT ROLE sys:dba TO user1;
```

下列範例會嘗試將循環相依性中的 sample_role3 授予 sample_role2。

```
CREATE ROLE sample_role3;
GRANT ROLE sample_role2 TO ROLE sample_role3;
GRANT ROLE sample_role3 TO ROLE sample_role2; -- fail
ERROR:  cannot grant this role, a circular dependency was detected between these roles
```

INSERT

主題

將新資料列插入資料表中。您可以使用 VALUES 語法插入單一資料列、使用 VALUES 語法插入多個資料列，或是一個或多個由查詢結果所定義的資料列 (INSERT INTO...SELECT)。

> **Note**
>
> 我們強烈鼓勵您使用 COPY 命令來載入大量資料。使用個別 INSERT 陳述式填入資料表的速度可能會相當慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用 INSERT INTO SELECT 或 CREATE TABLE AS 來改善效能。如需使用 COPY 命令載入資料表的相關資訊，請參閱 載入資料。

> **Note**
>
> 單一 SQL 陳述式的大小上限為 16 MB。

語法

```
INSERT INTO table_name [ ( column [, ...] ) ]
{DEFAULT VALUES |
VALUES ( { expression | DEFAULT } [, ...] )
[, ( { expression | DEFAULT } [, ...] )
[, ...] ] |
query }
```

參數

table_name

暫時性或持久性資料表。只有資料表的擁有者，或具有資料表 INSERT 權限的使用者可以插入資料列。如果您使用 query 子句插入資料列，則必須具有查詢中所指名資料表的 SELECT 權限。

> **Note**
>
> 使用 INSERT (外部資料表)，將 SELECT 查詢結果插入外部目錄上的現有資料表。如需詳細資訊，請參閱 INSERT (外部資料表)。

欄位

您可以將值插入資料表的一個或多個資料欄。您可以依任意順序列出目標資料欄名稱。若您未指定資料欄清單，則要插入的值必須依照 CREATE TABLE 陳述式中宣告的順序對應資料表資料欄。若要插入的值數目少於資料表中的資料欄數，則會在前 n 個資料欄中載入。

任何未在 INSERT 陳述式中列出的資料欄中都會載入宣告的預設值或 null 值 (隱含或明確)。

DEFAULT VALUES

若在建立資料表時對資料表中的資料欄指派預設值，請使用這些關鍵字插入完全由預設值組成的資料列。若有任何資料欄未包含預設值，則會將 null 插入這些資料欄。若有任何資料欄宣告 NOT NULL，則 INSERT 陳述式會傳回錯誤。

VALUES

使用此關鍵字插入一個或多個資料列，每列包含一個或多個值。每列的 VALUES 清單必須符合資料欄清單。若要插入多個資料列，請使用逗號分隔符號分隔每份表達式清單。切勿重複 VALUES 關鍵字。多列 INSERT 陳述式的所有 VALUES 清單必須包含相同數目的值。

運算式

單一值，或判斷值為單一值的表達式。每個值都必須與其插入所在資料欄的資料類型相容。若值的資料類型與資料欄的宣告資料類型不相符，則會在可能的情況下自動轉換成相容的資料類型。例如：

- 小數值 1.1 插入 INT 資料欄時會是 1。
- 小數值 100.8976 插入 DEC(5,2) 資料欄時會是 100.90。

您可以在表達式中包含類型轉換語法，藉此明確將值轉換成相容的資料類型。例如，若資料表 T1 中的資料欄 COL1 是 CHAR(3) 資料欄：

```
insert into t1(col1) values('Incomplete'::char(3));
```

此陳述式會將值 Inc 插入資料欄。

若是單一資料列 INSERT VALUES 陳述式，您可以使用純量子查詢做為表達式。子查詢的結果會插入適當的資料欄中。

> **Note**
>
> 多資料列 INSERT VALUES 陳述式中不支援使用子查詢做為表達式。

DEFAULT

使用此關鍵字可依資料表建立時所定義，插入資料欄的預設值。若資料欄沒有預設值，則會插入 null。若具有 NOT NULL 限制條件的資料欄在 CREATE TABLE 陳述式中沒有明確指定的預設值，則無法將預設值插入該資料欄。

query

藉由定義任何查詢將一個或多個資料列插入資料表。查詢產生的所有資料列都會插入資料表中。查詢必須傳回與資料表中資料欄相容的資料欄清單，但資料欄名稱不需相符。

使用須知

> **Note**
>
> 我們強烈鼓勵您使用 COPY 命令來載入大量資料。使用個別 INSERT 陳述式填入資料表的速度可能會相當慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用 INSERT INTO SELECT 或 CREATE TABLE AS 來改善效能。如需使用 COPY 命令載入資料表的相關資訊，請參閱 載入資料。

所插入值的資料格式必須符合 CREATE TABLE 定義所指定的資料格式。

將大量新資料列插入資料表之後：

- 清空資料表以回收儲存空間和重新排序資料列。
- 分析資料表以更新查詢規劃器的統計資訊。

若插入 DECIMAL 資料欄的值超出指定的小數位數，則載入的值會適當捨入。例如，若將值 `20.259` 插入 DECIMAL(8,2) 欄，儲存的值會是 `20.26`。

您可以插入至 GENERATED BY DEFAULT AS IDENTITY 資料欄。您可以利用您提供的值，更新定義為 GENERATED BY DEFAULT AS IDENTITY 的資料欄。如需詳細資訊，請參閱 GENERATED BY DEFAULT AS IDENTITY。

INSERT 範例

TICKIT 資料庫中的 CATEGORY 資料表包含以下資料列：

```
 catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
     1 | Sports   | MLB       | Major League Baseball
     2 | Sports   | NHL       | National Hockey League
     3 | Sports   | NFL       | National Football League
     4 | Sports   | NBA       | National Basketball Association
     5 | Sports   | MLS       | Major League Soccer
     6 | Shows    | Musicals  | Musical theatre
     7 | Shows    | Plays     | All non-musical theatre
     8 | Shows    | Opera     | All opera and light opera
     9 | Concerts | Pop       | All rock and pop music concerts
    10 | Concerts | Jazz      | All jazz singers and bands
```

```
    11 | Concerts | Classical | All symphony, concerto, and choir concerts
(11 rows)
```

以類似 CATEGORY 資料表的結構描述建立 CATEGORY_STAGE 資料表，但定義資料欄的預設值：

```
create table category_stage
(catid smallint default 0,
catgroup varchar(10) default 'General',
catname varchar(10) default 'General',
catdesc varchar(50) default 'General');
```

以下 INSERT 陳述式會選取 CATEGORY 資料表中的所有資料列，然後將它們插入 CATEGORY_STAGE 資料表。

```
insert into category_stage
(select * from category);
```

查詢前後的括號是選用的。

此命令會將新資料列插入 CATEGORY_STAGE 資料表，並且依序為每個資料欄指定值：

```
insert into category_stage values
(12, 'Concerts', 'Comedy', 'All stand-up comedy performances');
```

您也可以插入結合特定值與預設值的新資料列：

```
insert into category_stage values
(13, 'Concerts', 'Other', default);
```

執行以下查詢來傳回插入的資料列：

```
select * from category_stage
where catid in(12,13) order by 1;

 catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
    12 | Concerts | Comedy  | All stand-up comedy performances
    13 | Concerts | Other   | General
(2 rows)
```

以下範例說明一些多資料列 INSERT VALUES 陳述式。第一個範例會在兩個資料列中插入特定 CATID 值，並且在兩個資料列的其他資料欄中插入預設值。

```
insert into category_stage values
(14, default, default, default),
(15, default, default, default);

select * from category_stage where catid in(14,15) order by 1;
 catid | catgroup | catname | catdesc
-------+----------+---------+---------
    14 | General  | General | General
    15 | General  | General | General
(2 rows)
```

下一個範例會插入包含不同的特定與預設值組合的三個資料列：

```
insert into category_stage values
(default, default, default, default),
(20, default, 'Country', default),
(21, 'Concerts', 'Rock', default);

select * from category_stage where catid in(0,20,21) order by 1;
 catid | catgroup | catname | catdesc
-------+----------+---------+---------
     0 | General  | General | General
    20 | General  | Country | General
    21 | Concerts | Rock    | General
(3 rows)
```

此範例中第一組 VALUES 產生的結果，與針對單一資料列 INSERT 陳述式指定 DEFAULT VALUES 的結果相同。

以下範例說明資料表有 IDENTITY 資料欄時的 INSERT 行為。首先，建立新的 CATEGORY 資料表版本，然後從 CATEGORY 將資料列插入其中：

```
create table category_ident
(catid int identity not null,
catgroup varchar(10) default 'General',
catname varchar(10) default 'General',
catdesc varchar(50) default 'General');
```

```
insert into category_ident(catgroup,catname,catdesc)
select catgroup,catname,catdesc from category;
```

請注意，您無法將特定整數值插入 CATID IDENTITY 資料欄。IDENTITY 資料欄的值會自動產生。

以下範例將示範，多資料列 INSERT VALUES 陳述式中無法使用子查詢做為表達式：

```
insert into category(catid) values
((select max(catid)+1 from category)),
((select max(catid)+2 from category));

ERROR: can't use subqueries in multi-row VALUES
```

下列範例會示範使用 WITH SELECT 子句將資料從 venue 資料表填入暫存資料表中的插入操作。如需 venue 資料表的相關資訊，請參閱 範本資料庫。

首先，建立暫存資料表 #venuetemp。

```
CREATE TABLE #venuetemp AS SELECT * FROM venue;
```

列出 #venuetemp 資料表中的資料列。

```
SELECT * FROM #venuetemp ORDER BY venueid;

venueid | venuename                | venuecity   | venuestate| venueseats
--------+--------------------------+-------------+-----------+------------
1        Toyota Park                Bridgeview    IL          0
2        Columbus Crew Stadium      Columbus      OH          0
3        RFK Stadium                Washington    DC          0
4        CommunityAmerica Ballpark  Kansas City   KS          0
5        Gillette Stadium           Foxborough    MA          68756
...
```

使用 WITH SELECT 子句在 #venuetemp 資料表中插入 10 個重複的資料列。

```
INSERT INTO #venuetemp (WITH venuecopy AS (SELECT * FROM venue) SELECT * FROM venuecopy
 ORDER BY 1 LIMIT 10);
```

列出 #venuetemp 資料表中的資料列。

```
SELECT * FROM #venuetemp ORDER BY venueid;
```

```
venueid | venuename                | venuecity  | venuestate| venueseats
--------+--------------------------+------------+-----------+------------
1       Toyota Park                Bridgeview   IL          0
1       Toyota Park                Bridgeview   IL          0
2       Columbus Crew Stadium      Columbus     OH          0
2       Columbus Crew Stadium      Columbus     OH          0
3       RFK Stadium                Washington   DC          0
3       RFK Stadium                Washington   DC          0
4       CommunityAmerica Ballpark  Kansas City  KS          0
4       CommunityAmerica Ballpark  Kansas City  KS          0
5       Gillette Stadium           Foxborough   MA          68756
5       Gillette Stadium           Foxborough   MA          68756
...
```

INSERT (外部資料表)

將 SELECT 查詢的結果插入外部目錄上的現有外部資料表，例如用於 AWS Glue AWS Lake Formation、或 Apache Hive 中繼存放區。使用與「建立外部結構描述」命令相同的 AWS Identity and Access Management (IAM) 角色與外部目錄和 Amazon S3 互動。

對於未分割的資料表，INSERT (外部資料表) 命令會根據指定的資料表屬性和檔案格式，將資料寫入資料表中定義的 Amazon S3 位置。

針對分割的資料表，INSERT (外部資料表) 會根據資料表中指定的分割區索引鍵，將資料寫入 Amazon S3 位置。它也會在 INSERT 操作完成之後，自動在外部目錄中註冊新的分割區。

您不能在交易區塊 (BEGIN ... END) 內執行 INSERT (外部資料表)。如需交易的相關資訊，請參閱 可序列化隔離。

語法

```
INSERT INTO external_schema.table_name
{ select_statement }
```

參數

external_schema.table_name

現有外部資料結構描述的名稱和要插入的目標外部資料表。

select_statement

透過定義任何查詢，將一或多個列插入外部資料表的陳述式。所有查詢產生的列都會根據資料表定義，以文字或 Parquet 的格式被寫入 Amazon S3。查詢必須傳回與外部資料表中的欄位資料類型相容的欄位清單。不過，欄位名稱不一定要相符。

使用須知

SELECT 查詢中的欄位數必須與資料欄位和分割區欄位的總和相同。每個資料欄的位置和資料類型必須與外部資料表的資料類型相符。分割區欄位的位置必須在 SELECT 查詢的結尾，與它們在 CREATE EXTERNAL TABLE 命令定義的順序相同。欄位名稱不一定要相符。

在某些情況下，您可能想要在 AWS Glue 資料目錄或 Hive 中繼存放區上執行 INSERT (外部資料表) 命令。在這種情況下 AWS Glue，用於建立外部結構描述的 IAM 角色必須在 Amazon S3 和上同時具有讀取和寫入許可 AWS Glue。如果您使用 AWS Lake Formation 目錄，則此 IAM 角色將成為新 Lake Formation 表格的擁有者。此 IAM 角色至少必須具有下列許可：

- 在外部資料表上的 SELECT、INSERT、UPDATE 許可
- 外部資料表 Amazon S3 路徑上的資料位置許可

為了確保檔案名稱是唯一的，Amazon Redshift 依預設會針對每個上傳到 Amazon S3 的檔案名稱使用下列格式。

*<date>*_*<time>*_*<microseconds>*_*<query_id>*_*<slice-number>*_part_*<part-number>*.*<format>*.

例如，20200303_004509_810669_1007_0001_part_00.parquet。

執行 INSERT (外部資料表) 命令時，請考慮下列事項：

- 不支援非 PARQUET 或 TEXTFILE 格式的外部資料表。
- 此命令支援現有的資料表屬性，如 'write.parallel'、'write.maxfilesize.mb'、'compression_type' 和 'serialization.null.format'。若要更新這些值，請執行 ALTER TABLE SET TABLE PROPERTIES 命令。
- 'numRows' 資料表屬性會朝向 INSERT 操作結尾自動更新。如果資料表屬性不是由 CREATE EXTERNAL TABLE AS 操作建立，則資料表屬性必須已定義或新增到資料表。
- LIMIT 子句在外部的 SELECT 查詢中不受支援。請改用巢狀 LIMIT 子句。

- 您可以使用此 STL_UNLOAD_LOG 資料表來追蹤每個 INSERT (外部資料表) 操作寫入 Amazon S3 的檔案。
- Amazon Redshift 僅支援對 INSERT (外部資料表) 使用 Amazon S3 標準加密。

INSERT (外部資料表) 範例

下列範例會將 SELECT 陳述式的結果插入外部資料表。

```
INSERT INTO spectrum.lineitem
SELECT * FROM local_lineitem;
```

下列範例會使用靜態分割，將 SELECT 陳述式的結果插入已分割的外部資料表。分割區欄位是在 SELECT 陳述式中以硬式編碼寫成。分割區欄位必須位於查詢結尾。

```
INSERT INTO spectrum.customer
SELECT name, age, gender, 'May', 28 FROM local_customer;
```

下列範例會使用動態分割，將 SELECT 陳述式的結果插入已分割的外部資料表。分割區欄位並非硬式編碼。若已新增新的分割區，則資料會自動新增至現有的分割區資料夾或新的資料夾。

```
INSERT INTO spectrum.customer
SELECT name, age, gender, month, day FROM local_customer;
```

LOCK

限制對資料庫資料表的存取。此命令只有在交易區塊內執行時才有意義。

LOCK 命令會以 "ACCESS EXCLUSIVE" 模式取得資料表層級鎖定，必要時會等待任何衝突鎖定解除。若以此方式明確鎖定資料表，則從其他交易或工作階段嘗試時，會造成資料表的讀取和寫入等待。若某位使用者建立明確的資料表鎖定，則會暫時阻止其他使用者從該資料表選取資料或將資料載入其中。鎖定會在包含 LOCK 命令的交易完成時解除。

參考資料表的命令 (例如寫入操作) 會以隱含方式取得限制較少的資料表鎖定。例如，若使用者嘗試從資料表讀取資料時，有另一位使用者正在更新資料表，則讀取的資料會是已遞交資料的快照 (在某些情況下，若查詢違反可序列化隔離規則，則會停止)。請參閱管理並行寫入操作。

某些 DDL 操作 (例如 DROP TABLE 和 TRUNCATE) 會建立獨佔鎖定。這些操作會阻止資料讀取。

若發生鎖定衝突，Amazon Redshift 會顯示錯誤訊息，提醒進行發生衝突之交易的使用者。收到鎖定衝突的交易會停止。每次發生鎖定衝突時，Amazon Redshift 都會在 STL_TR_CONFLICT 資料表中寫入一項記錄。

語法

```
LOCK [ TABLE ] table_name [, ...]
```

參數

TABLE

選用的關鍵字。

table_name

要鎖定的資料表名稱。您可以使用逗號分隔的資料表名稱清單鎖定多個資料表。不過您無法鎖定檢視。

範例

```
begin;

lock event, sales;

...
```

MERGE

有條件地將來源資料表中的資料列合併到目標資料表中。傳統上，這只能透過單獨使用多個插入、更新或刪除陳述式來實現。如需使用 MERGE 讓您合併操作的相關資訊，請參閱 UPDATE、DELETE 和 INSERT。

語法

```
MERGE INTO target_table
USING source_table [ [ AS ] alias ]
ON match_condition
[ WHEN MATCHED THEN { UPDATE SET col_name = { expr } [,...] | DELETE }
WHEN NOT MATCHED THEN INSERT [ ( col_name [,...] ) ] VALUES ( { expr } [, ...] ) |
```

```
REMOVE DUPLICATES ]
```

參數

target_table

將與 MERGE 陳述式合併的暫存或永久資料表。

source_table

提供要合併到 target_table 之資料列的暫存或永久資料表。source_table 也可以是 Spectrum 資料表。source_table 不能是檢視或子查詢。

alias

source_table 暫時替代名稱。

此為選用參數。以 AS 開頭的 alias 也是選用項目。

match_condition

指定來源資料表資料欄與目標資料表資料欄之間的相等述詞，這些資料欄的作用是判斷 source_table 中的資料列是否可與 target_table 中的資料列相符。如果符合條件，則 MERGE 會針對該資料列執行 matched_clause。否則，MERGE 會為該資料列執行 not_matched_clause。

WHEN MATCHED

指定當來源資料列與目標資料列之間的比對條件評估為 True 時，要執行的動作。您可以指定 UPDATE 動作或 DELETE 動作。

UPDATE

更新 target_table 中的相符資料列。只會更新 col_name 中您指定的值。

DELETE

刪除 target_table 中的相符資料列。

WHEN NOT MATCHED

指定當比對條件評估為 False 或未知時要執行的動作。您只能為此子句指定 INSERT 插入動作。

INSERT

將一個資料列插入 target_table。您可以依任意順序列出目標 col_name。如果您不提供任何 col_name 值，則預設順序是按所有資料表資料欄的宣告順序排序。

col_name

您要修改的一個或多個資料欄名稱。不要在指定目標資料欄時包含資料表名稱。

expr

定義 col_name 新值的運算式。

REMOVE DUPLICATES

指定 MERGE 命令以簡化模式執行。簡化模式有以下要求：

- target_table 和 source_table 必須具有相同數目的資料欄和相容的資料欄類型。
- 從 MERGE 命令省略 WHEN 子句和 UPDATE 和 INSERT 子句。
- 在 MERGE 命令中使用 REMOVE DUPLICATES 子句。

在簡化模式中，MERGE 會執行下列操作：

- target_table 中具有 source_table 中相符項目的資料列會更新，以符合 source_table 中的值。
- source_table 中沒有 target_table 中相符項目的資料列會插入到 target_table 中。
- 當 target_table 中的多個資料列與 source_table 中的相同資料列相符時，重複的資料列會移除。Amazon Redshift 會保留一個資料列並對其進行更新。與 source_table 中的資料列不相符的重複資料列保持不變。

使用 REMOVE DUPLICATES 可提供比使用 WHEN MATCHED 和 WHEN NOT MATCHED 更好的效能。如果 target_table 和 source_table 相容，而且您不需要在 target_table 中保留重複資料列，我們建議您使用 REMOVE DUPLICATES。

使用須知

- 若要執行 MERGE 陳述式，您必須是 source_table 和 target_table 的擁有者，或具有這些資料表的 SELECT 許可。此外，視 MERGE 陳述式中包含的操作而定，您可能必須擁有 target_table 的 UPDATE、DELETE 和 INSERT 許可。
- target_table 不能是系統資料表、目錄資料表或外部資料表。
- source_table 和 target_table 不能是相同資料表。
- 您無法在 MERGE 陳述式中使用 WITH 子句。
- target_table 的資料列無法比對 source_table 中的多個資料列。

請思考下列範例：

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (1, 'Bob'), (2, 'John');
INSERT INTO source VALUES (1, 'Tony'), (1, 'Alice'), (3, 'Bill');

MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN UPDATE SET id = source.id, name = source.name
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);
ERROR: Found multiple matches to update the same tuple.

MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN DELETE
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);
ERROR: Found multiple matches to update the same tuple.
```

在這兩個 MERGE 陳述式中，操作都會失敗，因為 source 資料表中有多個 ID 值為 1 的資料列。

- match_condition 和 expr 不能部分參考 SUPER 類型的資料欄。例如，如果您的 SUPER 類型物件是陣列或結構，就不能將該資料欄的個別元素用於 match_condition 或 expr，但您可以使用整個資料欄。

  請思考下列範例：

```
CREATE TABLE IF NOT EXISTS target (key INT, value SUPER);
CREATE TABLE IF NOT EXISTS source (key INT, value SUPER);

INSERT INTO target VALUES (1, JSON_PARSE('{"key": 88}'));
INSERT INTO source VALUES (1, ARRAY(1, 'John')), (2, ARRAY(2, 'Bill'));

MERGE INTO target USING source ON target.key = source.key
WHEN matched THEN UPDATE SET value = source.value[0]
WHEN NOT matched THEN INSERT VALUES (source.key, source.value[0]);
ERROR: Partial reference of SUPER column is not supported in MERGE statement.
```

  如需 SUPER 類型的相關資訊，請參閱 SUPER 類型。

- 如果 source_table 很大，則將 target_table 和 source_table 中的聯接資料欄定義為分佈索引鍵可以提高性能。
- 若要使用 REMOVE DUPLICATES 子句，您需要 target_table 的 SELECT、INSERT 和 DELETE 許可。

範例

下列範例會建立兩個資料表，然後對其執行 MERGE 作業，並更新目標資料表中的相符資料列，以及插入不相符的資料列。然後將另一個值插入到來源表資料表中，並執行另一個 MERGE 操作，這次會刪除相符的資料列並從來源資料表中插入新資料列。

首先建立並填入來源和目標資料表。

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (101, 'Bob'), (102, 'John'), (103, 'Susan');
INSERT INTO source VALUES (102, 'Tony'), (103, 'Alice'), (104, 'Bill');

SELECT * FROM target;
 id  |    name
-----+------------
 101 | Bob
 102 | John
 103 | Susan
(3 rows)

SELECT * FROM source;
 id  |    name
-----+------------
 102 | Tony
 103 | Alice
 104 | Bill
(3 rows)
```

接下來，將來源資料表合併到目標資料表，使用相符的資料列更新目標資料表，並從來源資料表插入不相符的資料列。

```
MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN UPDATE SET id = source.id, name = source.name
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);

SELECT * FROM target;
 id  |    name
-----+------------
 101 | Bob
 102 | Tony
```

```
 103 | Alice
 104 | Bill
(4 rows)
```

請注意，識別碼值為 102 和 103 的資料列會更新，以符合目標資料表中的名稱值。此外，ID 值為 104 且名稱值為 Bill 的新資料列會插入目標資料表中。

接下來，在來源資料表中插入新資料列。

```
INSERT INTO source VALUES (105, 'David');

SELECT * FROM source;
 id  |    name
-----+------------
 102 | Tony
 103 | Alice
 104 | Bill
 105 | David
(4 rows)
```

最後，執行合併操作，刪除目標資料表中的相符資料列，並插入不相符的資料列。

```
MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN DELETE
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);

SELECT * FROM target;
 id  |    name
-----+------------
 101 | Bob
 105 | David
(2 rows)
```

識別碼值為 102、103 和 104 的資料列會從目標資料表中刪除，而識別碼值為 105 且名稱值為 David 的新資料列會插入目標資料表中。

下列範例顯示使用 REMOVE DUPLICATES 子句的 MERGE 命令。

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));
```

```
INSERT INTO target VALUES (30, 'Tony'), (11, 'Alice'), (23, 'Bill');
INSERT INTO source VALUES (23, 'David'), (22, 'Clarence');

MERGE INTO target USING source ON target.id = source.id REMOVE DUPLICATES;

SELECT * FROM target;
id | name
---+------------
30 | Tony
11 | Alice
23 | David
22 | Clarence
(4 rows)
```

下列範例顯示使用 REMOVE DUPLICATES 子句的 MERGE 命令，如果 target_table 中的資料列與 source_table 中的資料列相符，則會從前者中移除重複的資料列。

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (30, 'Tony'), (30, 'Daisy'), (11, 'Alice'), (23, 'Bill'),
 (23, 'Nikki');
INSERT INTO source VALUES (23, 'David'), (22, 'Clarence');

MERGE INTO target USING source ON target.id = source.id REMOVE DUPLICATES;

SELECT * FROM target;
id | name
---+------------
30 | Tony
30 | Daisy
11 | Alice
23 | David
22 | Clarence
(5 rows)
```

執行 MERGE 後， target_table 中只有一個 ID 值為 23 的資料列。由於 source_table 中沒有 ID 值為 30 的資料列，所以 ID 值為 30 的兩個重複資料列會保留在 target_table 中。

另請參閱

INSERT, UPDATE, DELETE

PREPARE

準備陳述式供執行。

PREPARE 會建立預備陳述式。當 PREPARE 陳述式執行時，會剖析、重寫及規劃指定的陳述式 (SELECT、INSERT、UPDATE 或 DELETE)。接著對預備陳述式發出 EXECUTE 命令時，Amazon Redshift 可能會選擇先修改查詢執行計畫 (依據指定的參數值改善效能)，再執行預備陳述式。

語法

```
PREPARE plan_name [ (datatype [, ...] ) ] AS statement
```

參數

plan_name

給予此特定預備陳述式的任意名稱。此名稱在單一工作階段內必須是唯一的，且後續會用來執行或解除配置先前的預備陳述式。

datatype

預備陳述式的參數資料類型。若要在預備陳述式本身內參考參數，請使用 $1、$2，以此類推。

statement

任何 SELECT、INSERT、UPDATE 或 DELETE 陳述式。

使用須知

預備陳述式可採用參數：這些值會在陳述式執行時替換到其中。若要在預備陳述式中包含參數，請在 PREPARE 陳述式中提供資料類型清單，然後在要準備的陳述式本身內使用 $1、$2 等符號依位置參考參數。當執行陳述式時，在 EXECUTE 陳述式中指定這些參數的實際值。如需詳細資訊，請參閱EXECUTE。

預備陳述式只會在目前工作階段期間內存在。當工作階段結束時，就會捨棄預備陳述式，因此必須再次建立它才能再度使用。這也表示，單一預備陳述式無法供多個同步的資料庫用戶端使用；不過，每個用戶端可建立自己要使用的預備陳述式。預備陳述式可使用 DEALLOCATE 命令手動移除。

在使用單一工作階段執行大量類似的陳述式時，預備陳述式能獲得最大效能。如前所述，每次重頭執行預備陳述式時，Amazon Redshift 都會根據指定的參數值修改查詢執行計畫以提升效能。若要檢查 Amazon Redshift 為任何特定 EXECUTE 陳述式選擇的查詢執行計畫，請使用 EXPLAIN 命令。

如需 Amazon Redshift 為進行查詢最佳化所收集的查詢計畫和統計資訊的相關資訊，請參閱 ANALYZE 命令。

範例

建立臨時資料表、準備 INSERT 陳述式，然後執行它：

```
DROP TABLE IF EXISTS prep1;
CREATE TABLE prep1 (c1 int, c2 char(20));
PREPARE prep_insert_plan (int, char)
AS insert into prep1 values ($1, $2);
EXECUTE prep_insert_plan (1, 'one');
EXECUTE prep_insert_plan (2, 'two');
EXECUTE prep_insert_plan (3, 'three');
DEALLOCATE prep_insert_plan;
```

準備 SELECT 陳述式，然後執行它：

```
PREPARE prep_select_plan (int)
AS select * from prep1 where c1 = $1;
EXECUTE prep_select_plan (2);
EXECUTE prep_select_plan (3);
DEALLOCATE prep_select_plan;
```

另請參閱

DEALLOCATE, EXECUTE

REFRESH MATERIALIZED VIEW

重新整理具體化檢視。

當您建立具體化視觀表時，其內容會反映當時基礎資料庫資料表或資料表的狀態。即使應用程式對基底資料表中的資料進行變更，具體化檢視中的資料仍會維持不變。如要更新具體化視觀表中的資料，您可以隨時使用 REFRESH MATERIALIZED VIEW 陳述式。當您執行此陳述式時，Amazon Redshift 會識別在基底資料表或資料表中發生的變更，然後將這些變更套用到具體化視觀表。

如需具體化檢視的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

語法

```
REFRESH MATERIALIZED VIEW mv_name
```

參數

mv_name

要重新整理的具體化檢視名稱。

使用須知

只有具體化檢視的擁有者才能對該檢視執行 REFRESH MATERIALIZED VIEW 操作。此外，擁有者必須對基礎資料表具有 SELECT 權限，才能成功執行 REFRESH MATERIALIZED VIEW。

REFRESH MATERIALIZED VIEW 命令會以自己所屬的交易形式執行。遵循 Amazon Redshift 交易語意，以決定 REFRESH 命令可以看到基礎資料表中的哪些資料，或是當 REFRESH 命令進行變更時，在 Amazon Redshift 中執行的其他交易是否會顯示。

- 針對累加式具體化檢視，REFRESH MATERIALIZED VIEW 只會使用已遞交的基底資料表資料列。因此，如果重新整理操作在相同交易中的資料處理語言 (DML) 陳述式之後執行，則重新整理看不見該 DML 陳述式的變更。
- 針對具體化視觀表的完全重新整理，REFRESH MATERIALIZED VIEW 會根據一般 Amazon Redshift 交易語意，查看重新整理交易可見的所有基底資料表資料列。
- 根據輸入引數類型，Amazon Redshift 仍支援具有特定輸入引數類型之下列函數的具體化檢視增量重新整理：DATE (時間戳記)、DATE_PART (日期、時間、間隔、time-tz)、DATE_TRUNC (時間戳記、間隔)。
- 基礎資料表位於資料共用中的具體化視觀表支援累加式重新整理。

Amazon Redshift 中的某些操作會和具體化視觀表互動。其中部份作業可能會強制 REFRESH MATERIALIZED VIEW 作業完全重新計算具體化檢視，即使定義具體化檢視的查詢只會使用適用於增量重新整理的 SQL 功能。例如：

- 如果具體化檢視並未重新整理，則背景 vacuum 操作可能會遭到封鎖。在內部定義的閾值期間後，便會允許執行 vacuum 操作。發生此 vacuum 操作時，任何依存的具體化檢視都會標記為在下一次重新整理時重新進行運算 (即使這些具體化檢視為累加式也一樣)。如需 VACUUM 的詳細資訊，請參閱 VACUUM。如需事件和狀態變更的相關資訊，請參閱 STL_MV_STATE。
- 有些使用者在基底資料表上啟動的操作會強制具體化檢視在下一次執行 REFRESH 操作時重新運算。這類操作的範例：手動調用的 VACUUM、傳統調整大小、ALTER DISTKEY 操作，ALTER SORTKEY 操作和 TRUNCATE 操作。如需事件和狀態變更的相關資訊，請參閱 STL_MV_STATE。

資料命名中具體化視觀表的累加式重新整理

Amazon Redshift 支援在共用基礎資料表時，針對消費者資料保護中的具體化視圖進行自動和累加式重新整理。累加式重新整理是一項操作，Amazon Redshift 會識別基礎資料表或上次重新整理後發生的資料表中的變更，並僅更新具體化視觀表中的對應記錄。如需此行為的相關資訊，請參閱建立具體化視觀表。

增量重新整理的限制

對於使用下列任一 SQL 元素搭配查詢定義的具體化視觀表，Amazon Redshift 目前不支援累加式重新整理：

- OUTER JOIN (RIGHT、LEFT 或 FULL)。
- 集合操作：UNION、INTERSECT、EXCEPT、MINUS。
- UNION ALL，當這發生在子查詢和彙總函數中，或 GROUP BY 子句存在於查詢中時。
- 彙整函數：
  MEDIAN、PERCENTILE_CONT、LISTAGG、STDDEV_SAMP、STDDEV_POP、APPROXIMATE COUNT、APPROXIMATE PERCENTILE 及位元彙整函數。

  > **Note**
  >
  > 支援 COUNT、SUM、MIN、MAX 和 AVG 彙總函數。

- DISTINCT 彙整函數，例如 DISTINCT COUNT、DISTINCT SUM 等。
- 視窗函數。
- 使用暫存資料表進行查詢最佳化的查詢，例如最佳化通用子運算式。
- 子查詢
- 在定義具體化視觀表的查詢中，參照下列查詢中的格式的外部資料表。
  - Delta Lake
  - Hudi

  使用參照預覽軌跡上其他格式的外部資料表定義的具體化視觀表，支援累加式重新整理。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的建立預覽叢集。如需設定預覽工作群組的詳細資訊，請參閱 Amazon Redshift 管理指南中的建立預覽工作群組。
- 可變函數，例如日期時間函數，RANDOM 和非穩定的使用者定義函數。

- 如需零 ETL 整合的累加式重新整理相關限制，請參閱將零 ETL 整合與 Amazon Redshift 搭配使用時的考量事項。

如需有關具體化視觀表的相關資訊，包括具體化視觀表重新整理操作上的背景作業影響，例如 VACUUM，請參閱 使用須知。

範例

以下範例會重新整理 tickets_mv 具體化檢視。

```
REFRESH MATERIALIZED VIEW tickets_mv;
```

RESET

將組態參數的值還原為其預設值。

您可以重設單一指定參數，或一次還原所有參數。若要將參數設定為特定值，請使用 SET 命令。若要顯示參數的目前值，請使用 SHOW 命令。

語法

```
RESET { parameter_name | ALL }
```

以下陳述式會將工作階段內容變數的值設定為 NULL。

```
RESET { variable_name | ALL }
```

參數

parameter_name

要重設的參數名稱。如需更多有關參數的文件，請參閱 修改伺服器組態。

ALL

重設所有執行期參數，包括所有工作階段內容變數。

variable

要重設的變數名稱 如果要執行 RESET 的值是工作階段內容變數，Amazon Redshift 會將其設定為 NULL。

範例

以下範例會將 query_group 參數重設為其預設值：

```
reset query_group;
```

以下範例會將所有執行時間參數重設為其預設值。

```
reset all;
```

下列範例會重設內容變數。

```
RESET app_context.user_id;
```

REVOKE

從使用者或角色移除存取許可，例如建立、捨棄或更新資料表的許可。

您只能對使用 ON SCHEMA 語法的資料庫使用者和角色執行外部結構描述的 GRANT 或 REVOKE USAGE 許可。搭配使用 ON 外部結構描述時 AWS Lake Formation，您只能授與和撤銷 AWS Identity and Access Management (IAM) 角色的權限。如需許可的清單，請參閱語法。

若為預存程序，系統依預設會將 USAGE ON LANGUAGE plpgsql 許可授予 PUBLIC。在預設情況下，EXECUTE ON PROCEDURE 許可只會授予擁有者和超級使用者。

在 REVOKE 命令中指定要移除的許可。若要授予許可，請使用 GRANT 命令。

語法

```
REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | ALTER | TRUNCATE } [,...] |
 ALL [ PRIVILEGES ] }
ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]


REVOKE [ GRANT OPTION FOR ]
{ { CREATE | TEMPORARY | TEMP | ALTER } [,...] | ALL [ PRIVILEGES ] }
ON DATABASE db_name [, ...]
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

```
[ RESTRICT ]


REVOKE [ GRANT OPTION FOR ]
{ { CREATE | USAGE | ALTER } [,...] | ALL [ PRIVILEGES ] }
ON SCHEMA schema_name [, ...]
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]


REVOKE [ GRANT OPTION FOR ]
EXECUTE
    ON FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]


REVOKE [ GRANT OPTION FOR ]
{ { EXECUTE } [,...] | ALL [ PRIVILEGES ] }
    ON PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]


REVOKE [ GRANT OPTION FOR ]
USAGE
    ON LANGUAGE language_name [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

授予資料表的資料欄層級許可

下列語法適用於 Amazon Redshift 資料表和檢視上的資料欄層級許可。

```
REVOKE { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
 ( column_name [,...] ) }
     ON { [ TABLE ] table_name [, ...] }
     FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
     [ RESTRICT ]
```

撤銷 ASSUMEROLE 許可

以下語法可將 ASSUMEROLE 許可從具有指定角色的使用者和群組中撤銷。

```
REVOKE ASSUMEROLE
    ON { 'iam_role' [, ...]  | default | ALL }
    FROM { user_name | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
    FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL }
```

撤銷 Redshift Spectrum 使用 Lake Formation 的許可

以下是 Redshift Spectrum 與 Lake Formation 整合的語法。

```
REVOKE [ GRANT OPTION FOR ]
{ SELECT | ALL [ PRIVILEGES ] } ( column_list )
    ON EXTERNAL TABLE schema_name.table_name
    FROM { IAM_ROLE iam_role } [, ...]

REVOKE [ GRANT OPTION FOR ]
{ { SELECT | ALTER | DROP | DELETE | INSERT }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL TABLE schema_name.table_name [, ...]
    FROM { { IAM_ROLE iam_role } [, ...] | PUBLIC }

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | ALTER | DROP }  [, ...] | ALL [ PRIVILEGES ] }
    ON EXTERNAL SCHEMA schema_name [, ...]
    FROM { IAM_ROLE iam_role } [, ...]
```

撤銷資料共用許可

生產者端資料共用權限

以下是使用 REVOKE 移除使用者或角色的 ALTER 或 SHARE 權限的語法。權限已被撤銷的使用者無法再變更資料共用，或將使用權授予取用者。

```
REVOKE { ALTER | SHARE } ON DATASHARE datashare_name
 FROM { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
 [, ...]
```

以下是使用 REVOKE 移除取用者對資料共用的存取的語法。

```
REVOKE USAGE
 ON DATASHARE datashare_name
 FROM NAMESPACE 'namespaceGUID' [, ...] | ACCOUNT 'accountnumber' [ VIA DATA CATALOG ]
 [, ...]
```

以下是將資料共用使用權從 Lake Formation 帳戶撤銷的範例。

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '123456789012' VIA DATA CATALOG;
```

取用者端資料共用權限

以下是資料共用使用許可的 REVOKE 語法，用於特定資料庫或從資料共用建立的結構描述。從用 WITH PERMISSIONS 子句建立的資料庫撤銷使用權限，不會撤銷您授予使用者或角色的其他任何權限，包括授予基礎物件的物件層級權限。如果您重新授予該使用者或角色的使用權限，他們會保留他們在您撤銷使用權之前擁有的其他所有權限。

```
REVOKE USAGE ON { DATABASE shared_database_name [, ...] | SCHEMA shared_schema}
 FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

撤銷限定範圍權限

作用域權限可讓您將權限授與資料庫或結構描述中某個類型之所有物件的使用者或角色。具有範圍權限的使用者和角色對資料庫或結構描述中所有目前和 future 物件都具有指定的權限。

以下是撤銷使用者和角色之限定範圍權限的語法。如需有關範圍權限的詳細資訊，請參閱。限定範圍權限

```
REVOKE [ GRANT OPTION ]
{ CREATE | USAGE | ALTER } [,...] | ALL [ PRIVILEGES ] }
FOR SCHEMAS IN
DATABASE db_name
FROM { username | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ]
{ { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
 [, ...] } | ALL [PRIVILEGES] } }
FOR TABLES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username] | ROLE role_name} [, ...]

REVOKE [ GRANT OPTION ] { EXECUTE | ALL [ PRIVILEGES ] }
FOR FUNCTIONS IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name | } [, ...]

REVOKE [ GRANT OPTION ] { EXECUTE | ALL [ PRIVILEGES ] }
```

```
FOR PROCEDURES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name | } [, ...]

REVOKE [ GRANT OPTION ] USAGE
FOR LANGUAGES IN
{DATABASE db_name}
FROM { username | ROLE role_name } [, ...]
```

請注意，作用域權限不會區分函數和程序的權限。例如，下列陳述式會從結構描述Sales_schema中撤銷函數和程序bob的EXECUTE權限。

```
REVOKE EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema FROM bob;
```

撤銷機器學習許可

以下是 Amazon Redshift 上的機器學習模型許可語法。

```
REVOKE [ GRANT OPTION FOR ]
    CREATE MODEL FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
    [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
    { EXECUTE | ALL [ PRIVILEGES ] }
    ON MODEL model_name [, ...]

    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
    [ RESTRICT ]
```

撤銷角色許可

以下是在 Amazon Redshift 上使用撤銷角色許可的語法。

```
REVOKE [ ADMIN OPTION FOR ] { ROLE role_name } [, ...] FROM { user_name } [, ...]
```

```
REVOKE { ROLE role_name } [, ...] FROM { ROLE role_name } [, ...]
```

以下是在 Amazon Redshift 上對角色撤銷系統許可的語法。

```
REVOKE
  {
```

```
    { CREATE USER | DROP USER | ALTER USER |
    CREATE SCHEMA | DROP SCHEMA |
    ALTER DEFAULT PRIVILEGES |
    ACCESS CATALOG |
    CREATE TABLE | DROP TABLE | ALTER TABLE |
    CREATE OR REPLACE FUNCTION | CREATE OR REPLACE EXTERNAL FUNCTION |
    DROP FUNCTION |
    CREATE OR REPLACE PROCEDURE | DROP PROCEDURE |
    CREATE OR REPLACE VIEW | DROP VIEW |
    CREATE MODEL | DROP MODEL |
    CREATE DATASHARE | ALTER DATASHARE | DROP DATASHARE |
    CREATE LIBRARY | DROP LIBRARY |
    CREATE ROLE | DROP ROLE
    TRUNCATE TABLE
    VACUUM | ANALYZE | CANCEL }[, ...]
  }
  | { ALL [ PRIVILEGES ] }
FROM { ROLE role_name } [, ...]
```

針對資料列層級安全性政策篩選撤銷解釋許可

以下語法可用來撤銷許可，以解釋 EXPLAIN 計畫中查詢的資料列層級安全性政策篩選。您可以使用 REVOKE 陳述式撤銷權限。

```
REVOKE EXPLAIN RLS FROM ROLE rolename
```

以下是授予許可以略過查詢資料列層級安全性政策的語法。

```
REVOKE IGNORE RLS FROM ROLE rolename
```

以下是從指定資料列層級安全性政策撤銷許可的語法。

```
REVOKE SELECT ON [ TABLE ] table_name [, ...]
            FROM RLS POLICY policy_name [, ...]
```

參數

GRANT OPTION FOR

只會撤銷對其他使用者授予指定許可的選項，而不會撤銷許可本身。您無法撤銷群組或 PUBLIC 的 GRANT OPTION。

SELECT

撤銷使用 SELECT 陳述式從資料表或檢視選取資料的許可。

INSERT

撤銷使用 INSERT 陳述式或 COPY 陳述式將資料載入資料表中的許可。

UPDATE

撤銷使用 UPDATE 陳述式更新資料表資料欄的許可。

DELETE

撤銷從資料表中刪除資料列的許可。

REFERENCES

撤銷建立外部索引鍵限制條件的許可。您應在被動參考資料表與主動參考資料表上撤銷此許可。

TRUNCATE

撤銷截斷資料表的許可。如果沒有此許可，只有資料表的擁有者或超級使用者可以截斷資料表。如需 TRUNCATE 命令的相關資訊，請參閱 the section called “TRUNCATE”。

ALL [PRIVILEGES]

從指定的使用者或群組一次撤銷所有可用許可。PRIVILEGES 關鍵字為選用。

ALTER

根據資料庫物件而定，將下列許可從使用者或使用者群組中撤銷：

- 對於資料表，ALTER 會撤銷修改資料表或檢視的許可。如需詳細資訊，請參閱 ALTER TABLE。
- 對於資料庫，ALTER 會撤銷修改資料庫的許可。如需詳細資訊，請參閱 ALTER DATABASE。
- 對於結構描述，ALTER 會授予修改結構描述的撤銷權。如需詳細資訊，請參閱 ALTER SCHEMA。
- 對於外部資料表，ALTER 會撤銷變更已啟用 Lake Formation AWS Glue Data Catalog 之表格的權限。此許可僅適用於使用 Lake Formation 時。

DROP

撤銷捨棄資料表的許可。此權限適用於 Amazon Redshift 以及為 Lake Formation 啟用的。 AWS Glue Data Catalog

ASSUMEROLE

從使用者、角色或具有指定角色的群組中撤銷執行 COPY、UNLOAD、EXTERNAL FUNCTION 或 CREATE MODEL 命令的許可。

ON [TABLE] table_name

撤銷資料表或檢視的指定許可。TABLE 關鍵字為選用。

ON ALL TABLES IN SCHEMA schema_name

撤銷所參考結構描述中所有資料表的指定許可。

(column_name [,...]) ON TABLE table_name

將 Amazon Redshift 資料表或檢視中指定資料欄的指定許可從使用者、群組或 PUBLIC 中撤銷。

(column_list) ON EXTERNAL TABLE schema_name.table_name

從所參考結構描述中 Lake Formation 資料表之指定欄上的 IAM 角色中撤銷指定許可。

ON EXTERNAL TABLE schema_name.table_name

從所參考結構描述中 Lake Formation 資料表上的 IAM 角色中撤銷指定許可。

ON EXTERNAL SCHEMA schema_name

從所參考結構描述上的 IAM 角色撤銷指定許可。

FROM IAM_ROLE iam_role

指出失去許可的 IAM 角色。

ROLE role_name

從指定的角色中撤銷許可。

GROUP group_name

從指定的使用者群組中撤銷許可。

PUBLIC

從所有使用者中撤銷指定許可。PUBLIC 代表永遠包含所有使用者的群組。個別使用者的許可是由授予 PUBLIC 的許可、授予使用者所屬之任何群組的許可，以及個別授予使用者的任何許可，三者加總所組成。

從 Lake Formation 外部資料表撤銷 PUBLIC 會從 Lake Formation everyone 群組撤銷該許可。

CREATE

根據資料庫物件而定，將下列許可從使用者或群組中撤銷：

- 若是資料庫，使用 CREATE 子句執行 REVOKE，可阻止使用者在資料庫內建立結構描述。

- 若是結構描述，使用 CREATE 子句執行 REVOKE，可阻止使用者在結構描述內建立物件。若要重新命名物件，使用者必須具有 CREATE 許可並擁有要重新命名的物件。

> **Note**
>
> 根據預設，所有使用者對 PUBLIC 結構描述都具有 CREATE 和 USAGE 許可。

TEMPORARY | TEMP

撤銷在指定資料庫中建立臨時資料表的許可。

> **Note**
>
> 根據預設，建立臨時資料表的許可是依使用者在 PUBLIC 群組中的自動成員資格授予。若要移除任何使用者建立臨時資料表的許可，請撤銷 PUBLIC 群組的 TEMP 權限，然後明確將建立臨時資料表的權限授予特定使用者或使用者群組。

ON DATABASE db_name

撤銷指定資料庫的許可。

USAGE

撤銷特定結構描述內物件的 USAGE 許可，讓使用者無法存取這些物件。對這些物件的特定動作必須另行撤銷 (例如函數的 EXECUTE 許可)。

> **Note**
>
> 根據預設，所有使用者對 PUBLIC 結構描述都具有 CREATE 和 USAGE 許可。

ON SCHEMA schema_name

撤銷指定結構描述的許可。您可以使用結構描述許可控制資料表的建立；資料庫的 CREATE 許可只能控制結構描述的建立。

RESTRICT

僅撤銷使用者直接授予的許可。這是預設行為。

EXECUTE ON PROCEDURE procedure_name

撤銷特定預存程序的 EXECUTE 許可。由於預存程序名稱可以過載，因此您必須包含程序的引數清單。如需詳細資訊，請參閱 命名預存程序。

EXECUTE ON ALL PROCEDURES IN SCHEMA procedure_name

撤銷所參考結構描述中所有程序的指定許可。

USAGE ON LANGUAGE language_name

撤銷某種語言的 USAGE 許可。若為 Python 使用者定義函數 (UDF)，請使用 `plpythonu`。若是 SQL UDF，請使用 `sql`。若為預存程序，請使用 `plpgsql`。

若要建立 UDF，您必須具有 SQL 或 `plpythonu` (Python) 的語言使用權許可。根據預設，USAGE ON LANGUAGE SQL 會授予 PUBLIC。不過，您必須將 USAGE ON LANGUAGE PLPYTHONU 明確授予特定使用者或群組。

若要撤銷 SQL 的使用權，請先從 PUBLIC 撤銷使用權。然後僅將 SQL 使用權授予獲得許可建立 SQL UDF 的特定使用者或群組。下列範例會撤銷 PUBLIC 的 SQL 使用權，然後將使用權授予使用者群組 `udf_devs`。

```
revoke usage on language sql from PUBLIC;
grant usage on language sql to group udf_devs;
```

如需詳細資訊，請參閱 UDF 安全與權限。

若要撤銷預存程序的使用權，請先從 PUBLIC 撤銷使用權。然後僅將 `plpgsql` 使用權授予獲得許可建立預存程序的特定使用者或群組。如需詳細資訊，請參閱 預存程序的安全和權限 。

FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL } [, ...]

指定要撤銷其權限的 SQL 命令。您可以指定 ALL 來撤銷 COPY、UNLOAD、EXTERNAL FUNCTION 和 CREATE MODEL 陳述式的許可。此子句僅適用於撤銷 ASSUMEROLE 許可。

ALTER

撤銷使用者或使用者群組的 ALTER 權限，該權限可讓未擁有資料共用的使用者或使用者群組修改資料共用。需要此許可才能在資料共用中新增或移除物件，或設定屬性 PUBLICACCESSIBLE。如需詳細資訊，請參閱 ALTER DATASHARE。

SHARE

撤銷使用者和使用者群組將取用者新增至資料共用的許可。撤銷此許可需要阻止特定取用者從其叢集存取資料共用。

ON DATASHARE datashare_name

授予參考資料共用的指定許可。

FROM 使用者名稱

指出失去許可的使用者。

FROM GROUP group_name

指出失去許可的使用者群組。

WITH GRANT OPTION

指出失去許可的使用者可以接著撤銷其他人的相同許可。您無法撤銷群組或 PUBLIC 的 WITH GRANT OPTION。

USAGE

對相同帳戶內的取用者帳戶或命名空間撤銷 USAGE 時，帳戶內的特定取用者帳戶或命名空間不可以以唯讀方式存取資料共用和資料共用的物件。

撤銷 USAGE 許可會撤銷取用者對資料共用的存取權。

FROM NAMESPACE 'clusternamespace GUID'

指出其中取用者失去資料共用許可的相同帳戶中的命名空間。命名空間會使用 128 位元英數字元的全域唯一識別碼 (GUID)。

FROM ACCOUNT 'accountnumber' [VIA DATA CATALOG]

指出其中取用者失去資料共用許可的其他帳戶中的帳戶編號。指定 'VIA DATA CATALOG' 表示您正在從 Lake Formation 帳戶中撤銷資料共用使用權。省略帳號代表您要從擁有叢集的帳戶撤銷。

ON DATABASE shared_database_name> [, ...]

在指定資料共用中建立的指定資料庫上撤銷指定使用許可。

ON SCHEMA shared_schema

在指定資料共用中建立的指定結構描述上撤銷指定許可。

FOR { SCHEMAS | TABLES | FUNCTIONS | PROCEDURES | LANGUAGES } IN

指定要撤銷權限的資料庫物件。IN 後面的參數定義撤銷權限的範圍。

CREATE MODEL

撤銷 CREATE MODEL 權限，以便在指定的資料庫中建立機器學習模型。

ON MODEL model_name

撤銷特定模型的 EXECUTE 許可。

ACCESS CATALOG

撤銷權限，以檢視角色可存取之物件的相關中繼資料。

[ADMIN OPTION FOR] { role } [, ...]

您從具有 WITH ADMIN OPTION 的指定使用者撤銷的角色。

FROM { role } [, ...]

您從中撤銷指定角色的角色。

使用須知

若要進一步了解 REVOKE 使用須知，請參閱 the section called “使用須知”。

範例

如需如何使用 REVOKE 的範例，請參閱 the section called “範例”。

使用須知

若要撤銷物件的權限，您必須符合下列條件之一：

- 身為物件擁有者。
- 身為超級使用者。
- 具有該物件和權限的授予權限。

例如，以下命令可讓使用者 HR 在 employees 資料表上執行 SELECT 命令，並對其他使用者授予和撤銷相同的權限。

```
grant select on table employees to HR with grant option;
```

HR 無法撤銷 SELECT 以外任何操作的權限，也無法撤銷 employees 資料表以外任何資料表的權限。

無論是使用 GRANT 或 REVOKE 命令設定物件權限，超級使用者都能存取所有物件。

PUBLIC 代表永遠包含所有使用者的群組。根據預設，PUBLIC 的所有成員對 PUBLIC 結構描述都具有 CREATE 和 USAGE 權限。若要限制 PUBLIC 結構描述上任何使用者的許可，您必須先撤銷 PUBLIC 結構描述上 PUBLIC 的所有許可，然後將權限授予特定使用者或群組。下列範例會控制 PUBLIC 結構描述中的資料表建立權限。

```
revoke create on schema public from public;
```

若要撤銷 Lake Formation 資料表的權限，則與該資料表外部結構描述相關聯的 IAM 角色就需具備撤銷外部資料表權限的許可。下列範例所建立的外部結構描述內含相關聯的 myGrantor IAM 角色。該 myGrantor IAM 角色具備撤銷他人許可的許可。REVOKE 命令會使用與外部結構描述相關聯的 myGrantor IAM 角色許可，撤銷 myGrantee IAM 角色的許可。

```
create external schema mySchema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myGrantor'
create external database if not exists;
```

```
revoke select
on external table mySchema.mytable
from iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

> **Note**
>
> 如果 IAM 角色也具有針對 Lake Formation 啟 AWS Glue Data Catalog 用的ALL權限，則不會撤銷該ALL權限。只有 SELECT 許可會遭撤銷。您可以在 Lake Formation 控制台中檢視 Lake Formation 許可。

撤銷 ASSUMEROL 許可的使用須知

下列是在 Amazon Redshift 中撤銷 ASSUMEROLE 權限的使用須知。

只有資料庫超級使用者才能撤銷使用者和群組的 ASSUMEROLE 權限。超級使用者始終保有 ASSUMEROLE 權限。

若要啟用使用者和群組的 ASSUMEROLE 權限，超級使用者要在叢集上執行一次下列陳述式。對使用者和群組授予 ASSUMEROLE 權限之前，超級使用者必須在叢集上執行一次下列陳述式。

```
revoke assumerole on all from public for all;
```

撤銷機器學習許可的使用須知

您無法直接授予或撤銷與 ML 函數相關的許可。ML 函數屬於 ML 模型，而且許可是透過模型控制的。相反地，您可以撤銷 ML 模型相關的許可。下列範例會示範如何從與模型 customer_churn 相關聯的所有使用者中撤銷執行許可。

```
REVOKE EXECUTE ON MODEL customer_churn FROM PUBLIC;
```

您也可以從 ML 模型 customer_churn 的使用者中撤銷所有許可。

```
REVOKE ALL on MODEL customer_churn FROM ml_user;
```

如果結構描述中有 ML 函數，則授予或撤銷 ML 函數相關的 EXECUTE 許可將會失敗，即使該 ML 函數已經具有透過 GRANT EXECUTE ON MODEL 取得的 EXECUTE 許可也是一樣。我們建議您在使用 CREATE MODEL 命令時，使用個別的結構描述，將 ML 函數本身保留在不同的結構描述中。下列範例示範如何執行此動作。

```
CREATE MODEL ml_schema.customer_churn
FROM customer_data
TARGET churn
FUNCTION ml_schema.customer_churn_prediction
IAM_ROLE default
SETTINGS (
 S3_BUCKET 'your-s3-bucket'
);
```

範例

以下範例會將 GUESTS 使用者群組在 SALES 資料表上的 INSERT 權限撤銷。此命令會使 GUESTS 的成員無法使用 INSERT 命令將資料載入 SALES 資料表。

```
revoke insert on table sales from group guests;
```

以下範例會將使用者 fred 對 QA_TICKIT 結構描述中所有資料表的 SELECT 權限撤銷。

```
revoke select on all tables in schema qa_tickit from fred;
```

下列範例會撤銷使用者 bobr 從檢視選取的權限。

```
revoke select on table eventview from bobr;
```

以下範例會將所有使用者在 TICKIT 資料庫中建立暫存資料表的權限撤銷。

```
revoke temporary on database tickit from public;
```

下列範例會撤銷使用者 user1 對 cust_profile 資料表中 cust_name 和 cust_phone 資料欄的 SELECT 權限。

```
revoke select(cust_name, cust_phone) on cust_profile from user1;
```

下列範例會撤銷 sales_group 群組對 cust_name 和 cust_phone 資料欄的 SELECT 權限，以及對 cust_profile 資料表中 cust_contact_preference 資料欄的 UPDATE 權限。

```
revoke select(cust_name, cust_phone), update(cust_contact_preference) on cust_profile
 from group sales_group;
```

下列範例顯示 ALL 關鍵字的使用方式，以撤銷 sales_admin 群組對資料表 cust_profile 中三個資料欄的 SELECT 和 UPDATE 權限。

```
revoke ALL(cust_name, cust_phone,cust_contact_preference) on cust_profile from group
 sales_admin;
```

以下範例會撤銷 user2 使用者對 cust_profile_vw 檢視中 cust_name 資料欄的 SELECT 權限。

```
revoke select(cust_name) on cust_profile_vw from user2;
```

從資料共用建立的資料庫撤銷 USAGE 權限的範例

下列範例會從 13b8833d-17c6-4f16-8fe4-1a018f5ed00d 命名空間撤銷對 salesshare 資料共用的存取權。

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
 '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

下列範例會撤銷 sales_db 上來自 Bob 的 USAGE 權限。

```
REVOKE USAGE ON DATABASE sales_db FROM Bob;
```

下列範例會撤銷 sales_schema 上來自 Analyst_role 的 USAGE 權限。

```
REVOKE USAGE ON SCHEMA sales_schema FROM ROLE Analyst_role;
```

撤銷限定範圍權限的範例

下列範例會將 Sales_db 資料庫中所有目前和未來結構描述的使用權從 Sales 角色撤銷。

```
REVOKE USAGE FOR SCHEMAS IN DATABASE Sales_db FROM ROLE Sales;
```

以下範例撤銷向使用者 alice 授予對 Sales_db 資料庫中所有目前和未來資料表的 SELECT 權限的能力。alice 保留對 Sales_db 中所有資料表的存取權。

```
REVOKE GRANT OPTION SELECT FOR TABLES IN DATABASE Sales_db FROM alice;
```

下列範例會將 Sales_schema 結構描述中函數的 EXECUTE 權限從 bob 角色撤銷。

```
REVOKE EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema FROM bob;
```

下列範例會從 Sales 角色撤銷 ShareDb 資料庫之 ShareSchema 結構描述中所有資料表的所有權限。當指定結構描述時，您也可以使用兩部分格式 database.schema 指定結構描述的資料庫。

```
REVOKE ALL FOR TABLES IN SCHEMA ShareDb.ShareSchema FROM ROLE Sales;
```

以下範例與前面的範例是相同的。您可以使用 DATABASE 關鍵字來指定結構描述資料庫，而不是使用兩部分格式。

```
REVOKE ALL FOR TABLES IN SCHEMA ShareSchema DATABASE ShareDb FROM ROLE Sales;
```

撤銷 ASSUMEROLE 權限的範例

以下是撤銷 ASSUMEROLE 權限的範例

超級使用者必須在叢集上執行一次下列陳述式，以啟用使用者和群組的 ASSUMEROLE 權限。

```
revoke assumerole on all from public for all;
```

下列陳述式會在所有操作的所有角色上撤銷使用者 reg_user1 的 ASSUMEROLE 權限。

```
revoke assumerole on all from reg_user1 for all;
```

撤銷 ROLE 權限的範例

下列範例會對 sample_role2 撤銷 sample_role1。

```
CREATE ROLE sample_role2;
GRANT ROLE sample_role1 TO ROLE sample_role2;
REVOKE ROLE sample_role1 FROM ROLE sample_role2;
```

下列範例會撤銷 user1 的系統權限。

```
GRANT ROLE sys:DBA TO user1;
REVOKE ROLE sys:DBA FROM user1;
```

下列範例會從 user1 撤銷 sample_role1 和 sample_role2。

```
CREATE ROLE sample_role1;
CREATE ROLE sample_role2;
GRANT ROLE sample_role1, ROLE sample_role2 TO user1;
REVOKE ROLE sample_role1, ROLE sample_role2 FROM user1;
```

下列範例會從 user1 撤銷具有 ADMIN OPTION 的 sample_role2。

```
GRANT ROLE sample_role2 TO user1 WITH ADMIN OPTION;
REVOKE ADMIN OPTION FOR ROLE sample_role2 FROM user1;
REVOKE ROLE sample_role2 FROM user1;
```

下列範例會從 sample_role5 撤銷 sample_role1 和 sample_role2。

```
CREATE ROLE sample_role5;
GRANT ROLE sample_role1, ROLE sample_role2 TO ROLE sample_role5;
REVOKE ROLE sample_role1, ROLE sample_role2 FROM ROLE sample_role5;
```

下列範例會撤銷 Sample_role1 的 CREATE SCHEMA 和 DROP SCHEMA 系統權限。

```
GRANT CREATE SCHEMA, DROP SCHEMA TO ROLE sample_role1;
REVOKE CREATE SCHEMA, DROP SCHEMA FROM ROLE sample_role1;
```

ROLLBACK

停止目前交易，並捨棄該交易所做的所有更新。

此命令會執行與 ABORT 命令相同的功能。

語法

```
ROLLBACK [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。預存程序內不支援使用此關鍵字。

TRANSACTION

選用的關鍵字。WORK 和 TRANSACTION 為同義詞，預存程序內不支援使用這兩個關鍵字。

如需在預存程序內使用 ROLLBACK 的相關資訊，請參閱管理交易。

範例

下列範例會建立資料表，然後開始交易，將資料插入資料表中。接著 ROLLBACK 命令就會轉返資料插入操作，而使資料表變成空白。

以下命令會建立名為 MOVIE_GROSS 的範例資料表：

```
create table movie_gross( name varchar(30), gross bigint );
```

下一個命令集會開始交易，將兩個資料列插入資料表中：

```
begin;

insert into movie_gross values ( 'Raiders of the Lost Ark', 23400000);

insert into movie_gross values ( 'Star Wars', 10000000 );
```

接著，以下命令會從資料表中選取資料，表示該資料已成功插入：

```
select * from movie_gross;
```

命令輸出會顯示這兩個資料列都已成功插入：

```
name           |  gross
-----------------------+----------
Raiders of the Lost Ark | 23400000
Star Wars               | 10000000
(2 rows)
```

現在這個命令會將資料變更轉返為交易開始時的狀態：

```
rollback;
```

從資料表選取資料現在會顯示空白資料表：

```
select * from movie_gross;

name | gross
------+-------
(0 rows)
```

SELECT

從資料表、檢視及使用者定義的函數中刪除資料列。

> **Note**
>
> 單一 SQL 陳述式的大小上限為 16 MB。

語法

```
[ WITH with_subquery [, ...] ]
SELECT
[ TOP number | [ ALL | DISTINCT ]
* | expression [ AS output_name ] [, ...] ]
[ FROM table_reference [, ...] ]
[ WHERE condition ]
[ [ START WITH expression ] CONNECT BY expression ]
```

```
[ GROUP BY expression [, ...] ]
[ HAVING condition ]
[ QUALIFY condition ]
[ { UNION | ALL | INTERSECT | EXCEPT | MINUS } query ]
[ ORDER BY expression [ ASC | DESC ] ]
[ LIMIT { number | ALL } ]
[ OFFSET start ]
```

主題

WITH 子句

WITH 子句是選用的子句，位於查詢中的 SELECT 前面。WITH 子句會定義一個或多個 common_table_expressions。每個通用資料表運算式 (CTE) 都會定義一個暫存資料表，與檢視定義類似。您可以在 FROM 子句中參考這些暫存資料表。這些資料表僅會在其所屬的查詢執行時使用。WITH 子句中的每個 CTE 都會指定資料表名稱、選用的資料欄名稱清單，以及判斷值為資料表的查詢表達式 (SELECT 陳述式)。當您在定義暫存資料表的相同查詢運算式的 FROM 子句中參考暫存資料表名稱時，CTE 是遞迴的。

WITH 子句子查詢是定義資料表時較有效率的方式，可在執行單一查詢的過程中使用。在所有任何情況下，於 SELECT 陳述式的本體中使用子查詢都可產生相同的結果，但 WITH 子句子查詢對於寫入和讀取來說可能較為簡單。參考多次的 WITH 子句子查詢會盡可能最佳化為通用子表達式；也就是說，或許可以評估 WITH 子查詢一次並重複使用其結果 (請注意，通用子表達式不限於 WITH 子句中所定義者)。

語法

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
```

其中 common_table_expression 可以是非遞迴或遞迴的。以下是非遞迴形式：

```
CTE_table_name [ ( column_name [, ...] ) ] AS ( query )
```

以下是 common_table_expression 的遞迴形式：

```
CTE_table_name (column_name [, ...] ) AS ( recursive_query )
```

參數

RECURSIVE

將查詢識別為遞迴 CTE 的關鍵字。如果 WITH 子句中定義的 common_table_expression 是遞迴的，則需要此關鍵字。即使 WITH 子句包含多個遞迴 CTE，您也只能指定一次 RECURSIVE 關鍵字 (緊接在 WITH 關鍵字之後)。一般而言，遞迴 CTE 是具有兩個部分的 UNION ALL 子查詢。

common_table_expression

定義可在 FROM 子句 中參照的暫存資料表，且僅在執行該資料表所屬的查詢期間使用此暫存資料表。

CTE_table_name

此臨時資料表的唯一名稱會定義 WITH 子句子查詢的結果。您無法在單一 WITH 子句內使用重複的名稱。每個子查詢都必須有可在 FROM 子句 中參考的資料表名稱。

column_name

WITH 子句子查詢的輸出資料欄名稱清單，以逗號分隔。指定的資料欄名稱數目必須等於或少於子查詢所定義的資料欄數目。對於非遞迴的 CTE 而言，column_name 子句是選擇性的。對於遞迴 CTE，column_name 清單是必要的。

query

Amazon Redshift 支援的任何 SELECT 查詢。請參閱SELECT。

recursive_query

由兩個 SELECT 子查詢組成的 UNION ALL 查詢：

- 第一個 SELECT 子查詢沒有對相同 CTE_table_name 的遞迴參考。這會傳回一個結果集，也就是遞迴的初始種子。此部分稱為初始成員或種子成員。
- 第二個 SELECT 子查詢會在其 FROM 子句中參考相同的 CTE_table_name。這就是所謂的遞迴成員。recursive_query 包含一個 WHERE 條件來結束 recursive_query。

使用須知

您可以在下列 SQL 陳述式中使用 WITH 子句：

- SELECT
- SELECT INTO
- CREATE TABLE AS
- CREATE VIEW
- DECLARE
- EXPLAIN
- INSERT INTO...SELECT
- PREPARE
- 更UPDATE (在 WHERE 子句子查詢中。您無法在子查詢中定義遞迴 CTE。遞迴 CTE 必須位於 UPDATE 子句之前。)
- DELETE

如果查詢的 FROM 子句包含 WITH 子句，但未參考 WITH 子句定義的任何資料表，則會忽略 WITH 子句，而查詢會照常執行。

WITH 子句子查詢定義的資料表只能在 WITH 子句開始的 SELECT 查詢範圍內參考。例如，您可以在 SELECT 清單、WHERE 子句或 HAVING 子句中，子查詢的 FROM 子句內參考這類資料表。您無法在子查詢中使用 WITH 子句，並於主查詢或其他子查詢的 FROM 子句內參考其資料表。此查詢模式會針對 WITH 子句資料表產生 `relation table_name doesn't exist` 形式的錯誤訊息。

您無法在 WITH 子句子查詢內指定另一個 WITH 子句。

您無法對 WITH 子句子查詢定義的資料表進行向前參考。例如，以下查詢會傳回錯誤訊息，因為資料表 W1 的定義中有對資料表 W2 的向前參考：

```
with w1 as (select * from w2), w2 as (select * from w1)
select * from sales;
```

```
ERROR:  relation "w2" does not exist
```

WITH 子句子查詢不一定包含 SELECT INTO 陳述式；不過您可以在 SELECT INTO 陳述式中使用 WITH 子句。

遞迴一般資料表表達式

遞迴 common table expression (CTE) 是參考其本身的 CTE。遞迴 CTE 在查詢階層式資料時非常有用，例如顯示員工與管理者之間責任關係的組織圖。請參閱範例：遞迴 CTE。

另一個常見的用途是多層級材料表，像是產品由許多元件組成，而每個元件本身也包含其他元件或次要組件時。

請務必在遞迴查詢的第二個 SELECT 子查詢中加入 WHERE 子句，以限制遞迴的深度。如需範例，請參閱範例：遞迴 CTE。否則，會發生類似以下內容的錯誤：

- `Recursive CTE out of working buffers.`
- `Exceeded recursive CTE max rows limit, please add correct CTE termination predicates or change the max_recursion_rows parameter.`

> **Note**
>
> `max_recursion_rows` 是一個參數，可設定遞迴 CTE 可傳回的最大資料列數，以防止無限遞迴迴圈。我們建議您不要將此值變更為大於預設值的值。這樣可以防止查詢中的無限遞迴佔用叢集中過多空間的問題。

您可以指定遞迴 CTE 結果的排序順序和限制。您可以在遞迴 CTE 的最終結果中包含群組依據和相異選項。

您無法在子查詢內指定 WITH RECURSIVE 子句。recursive_query 成員不能包含排序依據或限制子句。

範例

下列範例顯示包含 WITH 子句的最簡單查詢案例。名為 VENUECOPY 的 WITH 查詢會從 VENUE 資料表選取所有資料列。主查詢會接著從 VENUECOPY 選取所有資料列。VENUECOPY 資料表僅在此查詢期間存在。

```
with venuecopy as (select * from venue)
```

```
select * from venuecopy order by 1 limit 10;
```

```
 venueid |         venuename          |    venuecity    | venuestate | venueseats
---------+----------------------------+-----------------+------------+------------
1 | Toyota Park                | Bridgeview      | IL         |          0
2 | Columbus Crew Stadium      | Columbus        | OH         |          0
3 | RFK Stadium                | Washington      | DC         |          0
4 | CommunityAmerica Ballpark  | Kansas City     | KS         |          0
5 | Gillette Stadium           | Foxborough      | MA         |      68756
6 | New York Giants Stadium    | East Rutherford | NJ         |      80242
7 | BMO Field                  | Toronto         | ON         |          0
8 | The Home Depot Center      | Carson          | CA         |          0
9 | Dick's Sporting Goods Park | Commerce City   | CO         |          0
v     10 | Pizza Hut Park             | Frisco          | TX         |          0
(10 rows)
```

下列範例顯示 WITH 子句，它會產生兩個資料表，分別名為 VENUE_SALES 和 TOP_VENUES。第二個 WITH 查詢資料表會從第一個資料表選取。接著主查詢區塊的 WHERE 子句會包含限制 TOP_VENUES 資料表的子查詢。

```
with venue_sales as
(select venuename, venuecity, sum(pricepaid) as venuename_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
group by venuename, venuecity),

top_venues as
(select venuename
from venue_sales
where venuename_sales > 800000)

select venuename, venuecity, venuestate,
sum(qtysold) as venue_qty,
sum(pricepaid) as venue_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
and venuename in(select venuename from top_venues)
group by venuename, venuecity, venuestate
order by venuename;
```

```
       venuename       |   venuecity   | venuestate | venue_qty | venue_sales
-----------------------+---------------+------------+-----------+-------------
```

```
August Wilson Theatre   | New York City | NY         |      3187 |  1032156.00
Biltmore Theatre        | New York City | NY         |      2629 |   828981.00
Charles Playhouse       | Boston        | MA         |      2502 |   857031.00
Ethel Barrymore Theatre | New York City | NY         |      2828 |   891172.00
Eugene O'Neill Theatre  | New York City | NY         |      2488 |   828950.00
Greek Theatre           | Los Angeles   | CA         |      2445 |   838918.00
Helen Hayes Theatre     | New York City | NY         |      2948 |   978765.00
Hilton Theatre          | New York City | NY         |      2999 |   885686.00
Imperial Theatre        | New York City | NY         |      2702 |   877993.00
Lunt-Fontanne Theatre   | New York City | NY         |      3326 |  1115182.00
Majestic Theatre        | New York City | NY         |      2549 |   894275.00
Nederlander Theatre     | New York City | NY         |      2934 |   936312.00
Pasadena Playhouse      | Pasadena      | CA         |      2739 |   820435.00
Winter Garden Theatre   | New York City | NY         |      2838 |   939257.00
(14 rows)
```

以下兩個範例將示範根據 WITH 子句子查詢的資料表參考範圍規則。第一個查詢會執行，但第二個會失敗，並產生預期的錯誤。第一個查詢會在主查詢的 SELECT 清單內包含 WITH 子句子查詢。WITH 子句定義的資料表 (HOLIDAYS) 會在 SELECT 清單中子查詢的 FROM 子句中參考：

```
select caldate, sum(pricepaid) as daysales,
(with holidays as (select * from date where holiday ='t')
select sum(pricepaid)
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join date on sales.dateid=date.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;

caldate   | daysales | dec25sales
-----------+----------+------------
2008-12-25 | 70402.00 |   70402.00
2008-12-31 | 12678.00 |   70402.00
(2 rows)
```

第二個查詢會失敗，因為它會嘗試參考主查詢以及 SELECT 清單子查詢中的 HOLIDAYS 資料表。而主查詢參考超出範圍。

```
select caldate, sum(pricepaid) as daysales,
(with holidays as (select * from date where holiday ='t')
select sum(pricepaid)
```

```
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join holidays on sales.dateid=holidays.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;

ERROR:  relation "holidays" does not exist
```

範例：遞迴 CTE

以下是遞迴 CTE 的範例，此範例會傳回直接或間接向 John 報告的員工。遞迴查詢包含 WHERE 子句，可將遞迴深度限制為少於 4 個層級。

```
--create and populate the sample table
  create table employee (
  id int,
  name varchar (20),
  manager_id int
  );

  insert into employee(id, name, manager_id)  values
(100, 'Carlos', null),
(101, 'John', 100),
(102, 'Jorge', 101),
(103, 'Kwaku', 101),
(110, 'Liu', 101),
(106, 'Mateo', 102),
(110, 'Nikki', 103),
(104, 'Paulo', 103),
(105, 'Richard', 103),
(120, 'Saanvi', 104),
(200, 'Shirley', 104),
(201, 'Sofía', 102),
(205, 'Zhang', 104);

--run the recursive query
  with recursive john_org(id, name, manager_id, level) as
( select id, name, manager_id, 1 as level
  from employee
  where name = 'John'
  union all
  select e.id, e.name, e.manager_id, level + 1 as next_level
```

```
  from employee e, john_org j
  where e.manager_id = j.id and level < 4
  )
select distinct id, name, manager_id from john_org order by manager_id;
```

以下為查詢結果。

```
  id        name      manager_id
------+-----------+--------------
 101    John            100
 102    Jorge           101
 103    Kwaku           101
 110    Liu             101
 201    Sofía           102
 106    Mateo           102
 110    Nikki           103
 104    Paulo           103
 105    Richard         103
 120    Saanvi          104
 200    Shirley         104
 205    Zhang           104
```

以下是 John 的部門組織結構圖。

SELECT 清單

主題

- 語法
- 參數
- 使用須知
- 範例

SELECT 清單會指出您要查詢傳回的資料欄、函數及表達式的名稱。清單查詢的輸出。

如需 SQL 函數的相關資訊，請參閱 SQL 函數參考。如需運算式的相關資訊，請參閱條件式運算式。

語法

```
SELECT
```

```
[ TOP number ]
[ ALL | DISTINCT ] * | expression [ AS column_alias ] [, ...]
```

參數

TOP number

TOP 會採用正整數做為其引數，此引數定義傳回至用戶端的資料列數。TOP 子句的行為與 LIMIT 子句的行為相同。傳回的資料列數是固定的，但資料列集卻不是。若要傳回一致的資料列集，請搭配 ORDER BY 子句使用 TOP 或 LIMIT。

ALL

您未指定 DISTINCT 時，用來定義預設行為的備援關鍵字。SELECT ALL * 的意義與 SELECT * 相同 (選取所有資料欄的所有資料列並保留重複項目)。

DISTINCT

此選項會根據一個或多個資料欄中相符的值，從結果集中消除重複的資料列。

> **Note**
>
> 如果您的應用程式允許無效的外部索引鍵或主索引鍵，可能會導致查詢傳回不正確的結果。例如，如果主索引鍵資料行不包含所有唯一值，則 SELECT DISTINCT 查詢可能會傳回重複的資料列。如需詳細資訊，請參閱定義表格條件約束。

* (星號)

傳回資料表的整個內容 (所有資料欄和所有資料列)。

運算式

表達式是由查詢所參考資料表中的一個或多個資料欄構成。表達式可包含 SQL 函數。例如：

```
avg(datediff(day, listtime, saletime))
```

AS column_alias

資料欄的暫時名稱，會在最終結果集中使用。AS 關鍵字為選用。例如：

```
avg(datediff(day, listtime, saletime)) as avgwait
```

若您沒有為表達式指定非簡單資料欄名稱的別名，結果集將會套用預設名稱至該資料欄。

> **Note**
>
> 別名在目標清單中定義之後立即直接辨識。您可以在相同目標清單中後續定義的其他表達式後面使用別名。下列的範例示範了這一點。
>
> ```
> select clicks / impressions as probability, round(100 * probability, 1) as
>  percentage from raw_data;
> ```
>
> 側邊別名參考的好處在於，您在相同目標清單中建構更複雜的表達式時，不需要重複有別名的表達式。當 Amazon Redshift 剖析此類型參考時，會直接內嵌先前定義的別名。若 FROM 子句中定義了與先前具有別名之表達式同名的資料欄，則 FROM 子句中的資料欄優先順序較高。例如，在上方查詢中，若 raw_data 資料表中有名為 'probability' 的資料欄，目標清單中第二個表達式內的 'probability' 會參考該資料欄，而不是 'probability' 這個別名。

使用須知

TOP 是 SQL 延伸模組；它提供了 LIMIT 行為的替代方式。您無法在相同查詢中同時使用 TOP 和 LIMIT。

範例

下列範例會從 SALES 資料表中傳回 10 個資料列。雖然查詢使用 TOP 子句，但仍然會傳回無法預測的資料列集，因為沒有指定 ORDER BY 子句。

```
select top 10 *
from sales;
```

下列查詢具同等功能，但使用 LIMIT 子句而非 TOP 子句：

```
select *
from sales
limit 10;
```

下列範例會使用 TOP 子句從 SALES 資料表傳回前 10 列，並依 QTYSOLD 資料欄遞減排序。

```
select top 10 qtysold, sellerid
from sales
order by qtysold desc, sellerid;

qtysold | sellerid
--------+----------
8 |      518
8 |      520
8 |      574
8 |      718
8 |      868
8 |     2663
8 |     3396
8 |     3726
8 |     5250
8 |     6216
(10 rows)
```

下列範例會從 SALES 資料表傳回前兩個 QTYSOLD 和 SELLERID 值，並依 QTYSOLD 資料欄排序：

```
select top 2 qtysold, sellerid
from sales
order by qtysold desc, sellerid;

qtysold | sellerid
--------+----------
8 |      518
8 |      520
(2 rows)
```

下列範例顯示 CATEGORY 資料表中不同類別群組的清單：

```
select distinct catgroup from category
order by 1;

catgroup
----------
Concerts
Shows
Sports
(3 rows)
```

```
--the same query, run without distinct
select catgroup from category
order by 1;

catgroup
----------
Concerts
Concerts
Concerts
Shows
Shows
Shows
Sports
Sports
Sports
Sports
Sports
(11 rows)
```

下列範例會傳回 2008 年 12 月的不同週數組。如果沒有 DISTINCT 子句，陳述式會傳回 31 個資料列，或是針對每月的每一天傳回 1 列。

```
select distinct week, month, year
from date
where month='DEC' and year=2008
order by 1, 2, 3;

week | month | year
-----+-------+------
49 | DEC   | 2008
50 | DEC   | 2008
51 | DEC   | 2008
52 | DEC   | 2008
53 | DEC   | 2008
(5 rows)
```

FROM 子句

查詢中的 FROM 子句列出資料表參考 (資料表、檢視和子查詢)，此為選取資料的來源位置。若列出多個資料表參考，則必須在 FROM 子句或 WHERE 子句中使用適當的語法聯結資料表。若未指定聯結條件，則系統會將查詢當做交叉聯結 (笛卡兒乘積) 處理。

主題

語法

```
FROM table_reference [, ...]
```

其中 table_reference 是下列其中一項：

```
with_subquery_table_name [ table_alias ]
table_name [ * ] [ table_alias ]
( subquery ) [ table_alias ]
table_reference [ NATURAL ] join_type table_reference
   [ ON join_condition | USING ( join_column [, ...] ) ]
table_reference PIVOT (
   aggregate(expr) [ [ AS ] aggregate_alias ]
   FOR column_name IN ( expression [ AS ] in_alias [, ...] )
) [ table_alias ]
table_reference UNPIVOT [ INCLUDE NULLS | EXCLUDE NULLS ] (
   value_column_name
   FOR name_column_name IN ( column_reference [ [ AS ]
   in_alias ] [, ...] )
) [ table_alias ]
UNPIVOT expression AS value_alias [ AT attribute_alias ]
```

可選 table_alias 可用於為表和複雜表引用提供臨時名稱，如果需要，它們的列也可以使用，如下所示：

```
[ AS ] alias [ ( column_alias [, ...] ) ]
```

參數

with_subquery_table_name

WITH 子句 中子查詢所定義的資料表。

table_name

資料表或檢視的名稱。

alias

資料表或檢視的暫時替代名稱。必須為衍生自子查詢的資料表提供別名。在其他資料表參考中，別名是選用的。AS 關鍵字一律為選用。資料表別名提供了方便在查詢的其他部分中識別資料表的捷徑，例如 WHERE 子句。例如：

```
select * from sales s, listing l
where s.listid=l.listid
```

column_alias

資料表或檢視中資料欄的暫時替代名稱。

subquery

判斷值為資料表的查詢表達式。資料表只會在查詢期間存在，通常會為其命名或提供別名。不過，不需要別名。您也可以為衍生自子查詢的資料表定義資料欄名稱。當您想要將子查詢的結果與其他資料表聯結時，以及您想要在查詢中的其他位置選取或限制這些資料欄時，為資料欄指定別名就很重要。

子查詢可包含 ORDER BY 子句，但是，若未指定 LIMIT 或 OFFSET 子句，則此子句不一定有作用。

NATURAL

定義聯結，此聯結會自動使用兩個資料表中所有同名資料欄的配對做為聯結資料欄。不需要明確的聯結條件。例如，若 CATEGORY 和 EVENT 資料表都有名為 CATID 的資料欄，則這兩個資料表的 natural 聯結會是透過其 CATID 資料欄的聯結。

> **Note**
>
> 若已指定 NATURAL 聯結，但是要聯結的資料表中並沒有同名的資料欄配對，則查詢會預設為交叉聯結。

join_type

指定下列其中一種聯結類型：

- [INNER] JOIN
- LEFT [OUTER] JOIN
- RIGHT [OUTER] JOIN
- FULL [OUTER] JOIN
- CROSS JOIN

交叉聯結是沒有限定的聯結，會傳回兩個資料表的笛卡兒乘積。

內部和外部聯結為限定聯結。它們會以隱含方式 (在 natural 聯結中)、在 FROM 子句中使用 ON 或 USING 語法，或使用 WHERE 子句條件限定。

內部聯結只會根據聯結條件或聯結資料欄清單傳回相符的資料列。外部聯結會傳回對等內部聯結傳回的所有資料列，加上「左側」資料表和/或「右側」資料表的不相符資料列。左側資料表是最先列出的資料表，右側資料表是其次列出的資料表。不相符的資料列包含要填入輸出資料欄中空處的 NULL 值。

ON join_condition

聯結規格的類型，其中聯結資料欄會做為條件陳述，後面接著 ON 關鍵字。例如：

```
sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
```

USING (join_column [, ...])

聯結規格的類型，其中聯結資料欄會在括號內列出。若指定了多個聯結資料欄，則會以逗號分隔。USING 關鍵字必須放在清單前面。例如：

```
sales join listing
using (listid,eventid)
```

PIVOT

將輸出從資料列旋轉為資料欄，以便以易於閱讀的格式表示資料表資料。輸出會橫跨多個資料欄表示。PIVOT 類似於具使用彙總的 GROUP BY 查詢，並且會使用彙總運算式來指定輸出格式。但是，與 GROUP BY 相反，其結果會以資料欄傳回，而不是資料列。

如需示範如何使用 PIVOT 和 UNPIVOT 進行查詢的範例，請參閱 PIVOT 和 UNPIVOT 範例。

UNPIVOT

使用 UNPIVOT 將資料欄旋轉為資料列 — 運算子會將輸入資料表或查詢結果的結果欄轉換為列，以便於讀取輸出。UNPIVOT 會將其輸入資料欄的資料合併為兩個結果資料欄：名稱資料欄和值資料欄。名稱資料欄包含來自輸入中做為資料列項目的資料欄名稱。值資料欄包含來自輸入資料欄的值，例如彙總的結果。例如，各種類別中的項目計數。

使用 UNPIVOT (SUPER) 取消樞紐的物件 — 您可以執行取消樞紐的物件，其中運算式是參照另一個 FROM 子句項目的 SUPER 運算式。如需詳細資訊，請參閱 物件取消樞紐。它也有示範如何查詢半結構化資料的範例，例如 JSON 格式的資料。

使用須知

聯結資料欄必須採用可比較的資料類型。

NATURAL 或 USING 聯結只會針對中繼結果集內每個聯結資料欄配對保留一個。

使用 ON 語法的聯結則會保留其中繼結果集內的兩個聯結資料欄。

另請參閱WITH 子句。

PIVOT 和 UNPIVOT 範例

PIVOT 和 UNPIVOT 是 FROM 子句中的參數，這兩個參數會分別將查詢輸出從資料列旋轉為資料欄，以及將資料欄旋轉為資料列。這會以易於閱讀的格式呈現資料表式查詢結果。下列範例會使用測試資料和查詢來顯示如何使用這些參數。

如需這些參數與其他參數的相關資訊，請參閱 FROM 子句。

PIVOT 範例

設定範例資料表和資料，並用其來執行後續的範例查詢。

```
CREATE TABLE part (
    partname varchar,
    manufacturer varchar,
    quality int,
    price decimal(12, 2)
);
```

```
INSERT INTO part VALUES ('prop', 'local parts co', 2, 10.00);
INSERT INTO part VALUES ('prop', 'big parts co', NULL, 9.00);
INSERT INTO part VALUES ('prop', 'small parts co', 1, 12.00);

INSERT INTO part VALUES ('rudder', 'local parts co', 1, 2.50);
INSERT INTO part VALUES ('rudder', 'big parts co', 2, 3.75);
INSERT INTO part VALUES ('rudder', 'small parts co', NULL, 1.90);

INSERT INTO part VALUES ('wing', 'local parts co', NULL, 7.50);
INSERT INTO part VALUES ('wing', 'big parts co', 1, 15.20);
INSERT INTO part VALUES ('wing', 'small parts co', NULL, 11.80);
```

partname 上的 PIVOT 搭配 price 上的 AVG 彙總。

```
SELECT *
FROM (SELECT partname, price FROM part) PIVOT (
    AVG(price) FOR partname IN ('prop', 'rudder', 'wing')
);
```

此查詢結果為下列輸出。

```
  prop   |  rudder  |  wing
---------+----------+---------
 10.33   | 2.71     | 11.50
```

在上一個範例中，結果會轉換為資料欄。下列範例會顯示以資料列 (不是資料欄) 傳回平均價格的 GROUP BY 查詢。

```
SELECT partname, avg(price)
FROM (SELECT partname, price FROM part)
WHERE partname IN ('prop', 'rudder', 'wing')
GROUP BY partname;
```

此查詢結果為下列輸出。

```
 partname |  avg
----------+-------
 prop     | 10.33
 rudder   |  2.71
 wing     | 11.50
```

以 manufacturer 作為隱含資料欄的 PIVOT 範例。

```
SELECT *
FROM (SELECT quality, manufacturer FROM part) PIVOT (
    count(*) FOR quality IN (1, 2, NULL)
);
```

此查詢結果為下列輸出。

```
 manufacturer      | 1  | 2  | null
-------------------+----+----+------
 local parts co    | 1  | 1  |  1
 big parts co      | 1  | 1  |  1
 small parts co    | 1  | 0  |  2
```

未在 PIVOT 定義中參考的輸入資料表資料欄會隱含地新增至結果資料表。上一個範例中的 manufacturer 資料欄就是這種情況。此範例也會顯示 NULL 是 IN 運算子的有效值。

PIVOT 在上述範例中會傳回類似下列查詢的資訊，其中包括 GROUP BY。不同之處在於 PIVOT 傳回資料欄 2 的值 0 和製造商 small parts co。GROUP BY 查詢不包含對應的資料列。在大多數情況下，如果資料列沒有給定資料欄的輸入資料，PIVOT 會插入 NULL。但是，計數彙總不會傳回 NULL，0 是預設值。

```
SELECT manufacturer, quality, count(*)
FROM (SELECT quality, manufacturer FROM part)
WHERE quality IN (1, 2) OR quality IS NULL
GROUP BY manufacturer, quality
ORDER BY manufacturer;
```

此查詢結果為下列輸出。

```
 manufacturer        | quality | count
---------------------+---------+-------
 big parts co        |         |     1
 big parts co        |       2 |     1
 big parts co        |       1 |     1
 local parts co      |       2 |     1
 local parts co      |       1 |     1
 local parts co      |         |     1
 small parts co      |       1 |     1
 small parts co      |         |     2
```

PIVOT 運算子會接受彙總運算式上及 IN 運算子每個值上的選用別名。使用別名來自訂資料欄名稱。如果沒有彙總別名，則只會使用 IN 清單別名。否則，彙總別名會附加至資料欄名稱，並加上底線來分隔名稱。

```
SELECT *
FROM (SELECT quality, manufacturer FROM part) PIVOT (
    count(*) AS count FOR quality IN (1 AS high, 2 AS low, NULL AS na)
);
```

此查詢結果為下列輸出。

```
 manufacturer      | high_count  | low_count | na_count
-------------------+-------------+-----------+----------
 local parts co    |           1 |         1 |        1
 big parts co      |           1 |         1 |        1
 small parts co    |           1 |         0 |        2
```

設定下列範例資料表和資料，並用其來執行後續的範例查詢。資料表示飯店集合的預訂日期。

```
CREATE TABLE bookings (
    booking_id int,
    hotel_code char(8),
    booking_date date,
    price decimal(12, 2)
);

INSERT INTO bookings VALUES (1, 'FOREST_L', '02/01/2023', 75.12);
INSERT INTO bookings VALUES (2, 'FOREST_L', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (3, 'FOREST_L', '02/04/2023', 85.54);

INSERT INTO bookings VALUES (4, 'FOREST_L', '02/08/2023', 75.00);
INSERT INTO bookings VALUES (5, 'FOREST_L', '02/11/2023', 75.00);
INSERT INTO bookings VALUES (6, 'FOREST_L', '02/14/2023', 90.00);

INSERT INTO bookings VALUES (7, 'FOREST_L', '02/21/2023', 60.00);
INSERT INTO bookings VALUES (8, 'FOREST_L', '02/22/2023', 85.00);
INSERT INTO bookings VALUES (9, 'FOREST_L', '02/27/2023', 90.00);

INSERT INTO bookings VALUES (10, 'DESERT_S', '02/01/2023', 98.00);
INSERT INTO bookings VALUES (11, 'DESERT_S', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (12, 'DESERT_S', '02/04/2023', 85.00);
```

```
INSERT INTO bookings VALUES (13, 'DESERT_S', '02/05/2023', 75.00);
INSERT INTO bookings VALUES (14, 'DESERT_S', '02/06/2023', 34.00);
INSERT INTO bookings VALUES (15, 'DESERT_S', '02/09/2023', 85.00);

INSERT INTO bookings VALUES (16, 'DESERT_S', '02/12/2023', 23.00);
INSERT INTO bookings VALUES (17, 'DESERT_S', '02/13/2023', 76.00);
INSERT INTO bookings VALUES (18, 'DESERT_S', '02/14/2023', 85.00);

INSERT INTO bookings VALUES (19, 'OCEAN_WV', '02/01/2023', 98.00);
INSERT INTO bookings VALUES (20, 'OCEAN_WV', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (21, 'OCEAN_WV', '02/04/2023', 85.00);

INSERT INTO bookings VALUES (22, 'OCEAN_WV', '02/06/2023', 75.00);
INSERT INTO bookings VALUES (23, 'OCEAN_WV', '02/09/2023', 34.00);
INSERT INTO bookings VALUES (24, 'OCEAN_WV', '02/12/2023', 85.00);

INSERT INTO bookings VALUES (25, 'OCEAN_WV', '02/13/2023', 23.00);
INSERT INTO bookings VALUES (26, 'OCEAN_WV', '02/14/2023', 76.00);
INSERT INTO bookings VALUES (27, 'OCEAN_WV', '02/16/2023', 85.00);

INSERT INTO bookings VALUES (28, 'CITY_BLD', '02/01/2023', 98.00);
INSERT INTO bookings VALUES (29, 'CITY_BLD', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (30, 'CITY_BLD', '02/04/2023', 85.00);

INSERT INTO bookings VALUES (31, 'CITY_BLD', '02/12/2023', 75.00);
INSERT INTO bookings VALUES (32, 'CITY_BLD', '02/13/2023', 34.00);
INSERT INTO bookings VALUES (33, 'CITY_BLD', '02/17/2023', 85.00);

INSERT INTO bookings VALUES (34, 'CITY_BLD', '02/22/2023', 23.00);
INSERT INTO bookings VALUES (35, 'CITY_BLD', '02/23/2023', 76.00);
INSERT INTO bookings VALUES (36, 'CITY_BLD', '02/24/2023', 85.00);
```

在此查詢範例中，預訂記錄會計算為每週的總數。每週的結束日期會變成資料欄名稱。

```
SELECT * FROM
    (SELECT
       booking_id,
       (date_trunc('week', booking_date::date) + '5 days'::interval)::date as enddate,
       hotel_code AS "hotel code"
FROM bookings
) PIVOT (
    count(booking_id) FOR enddate IN ('2023-02-04','2023-02-11','2023-02-18')
);
```

此查詢結果為下列輸出。

```
 hotel code | 2023-02-04  | 2023-02-11 | 2023-02-18
------------+-------------+------------+----------
 FOREST_L   |           3 |          2 |        1
 DESERT_S   |           4 |          3 |        2
 OCEAN_WV   |           3 |          3 |        3
 CITY_BLD   |           3 |          1 |        2
```

Amazon Redshift 不支援使用 CROSSTAB 在多個資料欄上轉移。但是，您可以使用類似於 PIVOT 的彙總方式將資料列資料變更為資料欄，並使用類似於以下內容的查詢。這會使用與前一個範例相同的預訂範例資料。

```
SELECT
  booking_date,
  MAX(CASE WHEN hotel_code = 'FOREST_L' THEN 'forest is booked' ELSE '' END) AS
 FOREST_L,
  MAX(CASE WHEN hotel_code = 'DESERT_S' THEN 'desert is booked' ELSE '' END) AS
 DESERT_S,
  MAX(CASE WHEN hotel_code = 'OCEAN_WV' THEN 'ocean is booked' ELSE '' END)  AS
 OCEAN_WV
FROM bookings
GROUP BY booking_date
ORDER BY booking_date asc;
```

範例查詢結果會顯示預訂日期，並列在指出已預訂飯店的短語旁邊。

```
 booking_date  | forest_l         | desert_s         | ocean_wv
---------------+------------------+------------------+--------------------
 2023-02-01    | forest is booked | desert is booked |  ocean is booked
 2023-02-02    | forest is booked | desert is booked |  ocean is booked
 2023-02-04    | forest is booked | desert is booked |  ocean is booked
 2023-02-05    |                  | desert is booked |
 2023-02-06    |                  | desert is booked |
```

以下是 PIVOT 的使用須知：

- PIVOT 可套用至資料表、子查詢和通用資料表運算式 (CTE)。PIVOT 無法套用至任何 JOIN 運算式、遞迴 CTE、PIVOT 或 UNPIVOT 運算式。SUPER 非巢狀運算式和 Redshift Spectrum 巢狀資料表也不支援。
- PIVOT 支援 COUNT、SUM、MIN、MAX 和 AVG 彙總函數。

- PIVOT 彙總運算式必須是受支援彙總函式的呼叫。不支援彙總頂端的複雜運算式。彙總引數不能包含對 PIVOT 輸入資料表以外資料表的參考。也不支援父查詢的相關參考。彙總參數可以包含子查詢。這些可以在內部或 PIVOT 輸入資料表上相互關聯。
- PIVOT IN 清單值不能是資料欄參考或子查詢。每個值必須是與 FOR 資料欄參考相容的類型。
- 如果 IN 清單值沒有別名，則 PIVOT 會產生預設資料欄名稱。對於常數 IN 值，如 'abc' 或 5，預設資料欄名稱是常值本身。對於任何複雜的運算式，資料欄名稱都是標準的 Amazon Redshift 預設名稱，例如 ?column?.

UNPIVOT 範例

設定範例資料，並用其來執行後續的範例。

```
CREATE TABLE count_by_color (quality varchar, red int, green int, blue int);

INSERT INTO count_by_color VALUES ('high', 15, 20, 7);
INSERT INTO count_by_color VALUES ('normal', 35, NULL, 40);
INSERT INTO count_by_color VALUES ('low', 10, 23, NULL);
```

UNPIVOT 在輸入欄紅色、綠色和藍色上。

```
SELECT *
FROM (SELECT red, green, blue FROM count_by_color) UNPIVOT (
    cnt FOR color IN (red, green, blue)
);
```

此查詢結果為下列輸出。

```
 color | cnt
-------+-----
 red   |  15
 red   |  35
 red   |  10
 green |  20
 green |  23
 blue  |   7
 blue  |  40
```

依預設，輸入資料欄中的 NULL 值會略過，且不會產生結果資料列。

下列範例會顯示包括 INCLUDE NULLS 的 UNPIVOT。

```
SELECT *
FROM (
    SELECT red, green, blue
    FROM count_by_color
) UNPIVOT INCLUDE NULLS (
    cnt FOR color IN (red, green, blue)
);
```

以下為其輸出。

```
 color | cnt
-------+-----
 red   |  15
 red   |  35
 red   |  10
 green |  20
 green |
 green |  23
 blue  |   7
 blue  |  40
 blue  |
```

如果設定 INCLUDING NULLS 參數，NULL 輸入值會產生結果列。

以 quality 作為隱含資料欄的 The following query shows UNPIVOT。

```
SELECT *
FROM count_by_color UNPIVOT (
    cnt FOR color IN (red, green, blue)
);
```

此查詢結果為下列輸出。

```
 quality | color | cnt
---------+-------+-----
 high    | red   |  15
 normal  | red   |  35
 low     | red   |  10
 high    | green |  20
 low     | green |  23
 high    | blue  |   7
```

```
 normal  | blue  |  40
```

未在 UNPIVOT 定義中參考的輸入資料表資料欄會隱含地新增至結果資料表。在範例中，quality 資料欄就是這種情況。

下列範例顯示 IN 清單中具有值別名的 UNPIVOT。

```
SELECT *
FROM count_by_color UNPIVOT (
    cnt FOR color IN (red AS r, green AS g, blue AS b)
);
```

上述查詢結果為下列輸出。

```
 quality | color | cnt
---------+-------+-----
 high    | r     |  15
 normal  | r     |  35
 low     | r     |  10
 high    | g     |  20
 low     | g     |  23
 high    | b     |   7
 normal  | b     |  40
```

UNPIVOT 運算仔會接受每個 IN 清單值上的選用別名。每個別名都會提供每個 value 資料欄中的資料定義。

以下是 UNPIVOT 的使用須知。

- UNPIVOT 可套用至資料表、子查詢和通用資料表運算式 (CTE)。UNPIVOT 無法套用至任何 JOIN 運算式、遞迴 CTE、PIVOT 或 UNPIVOT 運算式。SUPER 非巢狀運算式和 Redshift Spectrum 巢狀資料表也不支援。
- UNPIVOT IN 清單必須只包含輸入資料表資料欄參考。IN 清單欄必須具有與之相容的通用類型。UNPIVOT 值資料欄具有這種通用類型。UNPIVOT 名稱資料欄的類型為 VARCHAR。
- 如果 IN 清單值沒有別名，UNPIVOT 會使用資料欄名稱做為預設值。

JOIN 範例

SQL JOIN 子句用於根據通用欄位，結合兩個或多個資料表中的資料。結果可能會或可能不會改變，具體取決於指定的聯結方法。如需 JOIN 子句語法的相關資訊，請參閱 參數。

下列範例會使用 TICKIT 範例資料中的資料。如需資料庫結構描述的相關資訊，請參閱 範本資料庫。若要了解如何載入範例資料，請參閱 Amazon Redshift 入門指南中的載入資料。

下列查詢是 LISTING 資料表和 SALES 資料表之間的內部聯結 (沒有 JOIN 關鍵字)，其中 LISTING 資料表中的 LISTID 是介於 1 和 5 之間。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示 LISTID 1、4 和 5 符合條件。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing, sales
where listing.listid = sales.listid
and listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

下列查詢是左側外部聯結。未在另一個資料表中找到相符項目時，左和右外部聯結會保留來自其中一個聯結資料表的值。左側和右側資料表分別是語法中最先和其次列出的資料表。NULL 值會用來填入結果集中的「空處」。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示，LISTID 2 和 3 並未產生任何銷售。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing left outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     2 | NULL   | NULL
     3 | NULL   | NULL
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

下列查詢是右側外部聯結。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示 LISTID 1、4 和 5 符合條件。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing right outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

下列查詢是完全聯結。未在另一個資料表中找到相符項目時，完全聯結會保留來自聯結資料表的值。左側和右側資料表分別是語法中最先和其次列出的資料表。NULL 值會用來填入結果集中的「空處」。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示，LISTID 2 和 3 並未產生任何銷售。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     2 | NULL   | NULL
     3 | NULL   | NULL
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

下列查詢是完全聯結。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。只有不會產生任何銷售 (ListID 2 和 3) 的資料列才會顯示在結果中。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
and (listing.listid IS NULL or sales.listid IS NULL)
group by 1
order by 1;

listid | price  |  comm
```

```
-------+--------+--------
     2 | NULL   | NULL
     3 | NULL   | NULL
```

下列範例是一個內部聯結搭配 ON 子句。在此情況下，不會傳回 NULL 資料列。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
where listing.listid between 1 and 5
group by 1
order by 1;

listid | price  |  comm
-------+--------+--------
     1 | 728.00 | 109.20
     4 |  76.00 |  11.40
     5 | 525.00 |  78.75
```

下列查詢是 LISTING 資料表和 SALES 資料表的交叉聯結或笛卡爾聯結，並且使用述詞來限制結果。此查詢會在 SALES 資料表和 LISTING 資料表中比對 LISTID 資料欄值，以找出兩個資料表中的 LISTID 1、2、3、4 和 5。結果顯示 20 個符合條件的資料列。

```
select sales.listid as sales_listid, listing.listid as listing_listid
from sales cross join listing
where sales.listid between 1 and 5
and listing.listid between 1 and 5
order by 1,2;

sales_listid | listing_listid
-------------+---------------
1            | 1
1            | 2
1            | 3
1            | 4
1            | 5
4            | 1
4            | 2
4            | 3
4            | 4
4            | 5
5            | 1
5            | 1
```

```
5            | 2
5            | 2
5            | 3
5            | 3
5            | 4
5            | 4
5            | 5
5            | 5
```

下列範例是兩個資料表之間的自然聯結。在這種情況下，兩個資料表中的 listid、sellerid、eventid 和 dateid 資料欄會具有相同的名稱和資料類型，因此會被用來作為聯結資料欄。結果限制為 5 個資料列。

```
select listid, sellerid, eventid, dateid, numtickets
from listing natural join sales
order by 1
limit 5;

listid | sellerid  | eventid | dateid | numtickets
-------+-----------+---------+--------+-----------
113    | 29704     | 4699    | 2075   | 22
115    | 39115     | 3513    | 2062   | 14
116    | 43314     | 8675    | 1910   | 28
118    | 6079      | 1611    | 1862   | 9
163    | 24880     | 8253    | 1888   | 14
```

下列範例是兩個資料表之間的聯結和 USING 子句。在這種情況下，資料欄 listid 和 eventid 會被用來作為聯結資料欄。結果限制為 5 個資料列。

```
select listid, listing.sellerid, eventid, listing.dateid, numtickets
from listing join sales
using (listid, eventid)
order by 1
limit 5;

listid | sellerid | eventid | dateid | numtickets
-------+----------+---------+--------+-----------
1      | 36861    | 7872    | 1850   | 10
4      | 8117     | 4337    | 1970   | 8
5      | 1616     | 8647    | 1963   | 4
5      | 1616     | 8647    | 1963   | 4
6      | 47402    | 8240    | 2053   | 18
```

以下查詢為 FROM 子句中兩個子查詢的內部聯結。查詢會尋找不同類別活動 (演奏會和表演) 的已售出和未售出票券數目。FROM 子句子查詢是資料表子查詢，可傳回多個資料欄和資料列。

```
select catgroup1, sold, unsold
from
(select catgroup, sum(qtysold) as sold
from category c, event e, sales s
where c.catid = e.catid and e.eventid = s.eventid
group by catgroup) as a(catgroup1, sold)
join
(select catgroup, sum(numtickets)-sum(qtysold) as unsold
from category c, event e, sales s, listing l
where c.catid = e.catid and e.eventid = s.eventid
and s.listid = l.listid
group by catgroup) as b(catgroup2, unsold)

on a.catgroup1 = b.catgroup2
order by 1;

catgroup1 |  sold  | unsold
----------+--------+--------
Concerts  | 195444 |1067199
Shows     | 149905 | 817736
```

WHERE 子句

WHERE 子句包含聯結資料表或套用述詞至資料表中資料欄的條件。資料表可以藉由在 WHERE 子句或 FROM 子句中使用適當的語法進行內部聯結。外部連結條件必須在 FROM 子句中指定。

語法

```
[ WHERE condition ]
```

條件

任何產生布林值結果的搜尋條件，例如，資料表資料欄的聯結條件或述詞。以下範例為有效的聯結條件：

```
sales.listid=listing.listid
sales.listid<>listing.listid
```

以下範例對於資料表中的資料欄是有效的條件：

```
catgroup like 'S%'
venueseats between 20000 and 50000
eventname in('Jersey Boys','Spamalot')
year=2008
length(catdesc)>25
date_part(month, caldate)=6
```

條件可分成簡單和複雜；若是複雜條件，您可以使用括號來隔離邏輯單位。在下列範例中，聯結條件會以括號包圍。

```
where (category.catid=event.catid) and category.catid in(6,7,8)
```

使用須知

您無法在 WHERE 子句中使用別名來參考選取清單表達式。

您無法限制 WHERE 子句中彙整函數的結果；請使用 HAVING 子句達成此目的。

WHERE 子句中限制的資料欄必須衍生自 FROM 子句中的資料表參考。

範例

以下查詢使用不同 WHERE 子句限制的組合，包括 SALES 和 EVENT 資料表的聯結條件、EVENTNAME 資料欄上的述詞，以及 STARTTIME 資料欄上的兩個述詞。

```
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
where sales.eventid = event.eventid
and eventname='Hannah Montana'
and date_part(quarter, starttime) in(1,2)
and date_part(year, starttime) = 2008
order by 3 desc, 4, 2, 1 limit 10;

eventname    |      starttime      |   costperticket   | qtysold
---------------+---------------------+-------------------+---------
Hannah Montana | 2008-06-07 14:00:00 |     1706.00000000 |       2
Hannah Montana | 2008-05-01 19:00:00 |     1658.00000000 |       2
Hannah Montana | 2008-06-07 14:00:00 |     1479.00000000 |       1
Hannah Montana | 2008-06-07 14:00:00 |     1479.00000000 |       3
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       1
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       2
Hannah Montana | 2008-06-07 14:00:00 |     1163.00000000 |       4
```

```
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       1
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       2
Hannah Montana | 2008-05-01 19:00:00 |      497.00000000 |       4
(10 rows)
```

WHERE 子句中 Oracle 樣式的外部聯結

為了提供 Oracle 相容性，Amazon Redshift 在 WHERE 子句聯結條件中支援 Oracle 外部聯結運算子 (+)。此運算子主要僅用於定義外部聯結條件；請勿嘗試在其他內容中使用它。大多數情況下，此運算子的其他用途都會加以忽略，且不會顯示任何訊息。

外部聯結會傳回對等內部聯結傳回的所有資料列，加上其中一個資料表或兩個資料表的不相符資料列。在 FROM 子句中，您可以指定左、右和完整外部聯結。在 WHERE 子句中，您只能指定左和右外部聯結。

若要對 TABLE1 和 TABLE2 進行外部聯結並從 TABLE1 傳回不相符的資料列 (左外部聯結)，請在 FROM 子句中指定 `TABLE1 LEFT OUTER JOIN TABLE2`，或在 WHERE 子句中對來自 TABLE2 的所有聯結資料欄套用 (+) 運算子。對於 TABLE1 中在 TABLE2 中沒有相符資料列的所有資料列，查詢的結果會對包含 TABLE2 中資料欄的任何選取清單表達式包含 null。

若要對 TABLE2 中在 TABLE1 中沒有相符資料列的所有資料列產生相同的行為，請在 FROM 子句中指定 `TABLE1 RIGHT OUTER JOIN TABLE2`，或在 WHERE 子句中對來自 TABLE1 的所有聯結資料欄套用 (+) 運算子。

基本語法

```
[ WHERE {
[ table1.column1 = table2.column1(+) ]
[ table1.column1(+) = table2.column1 ]
}
```

第一個條件相當於：

```
from table1 left outer join table2
on table1.column1=table2.column1
```

第二個條件相當於：

```
from table1 right outer join table2
on table1.column1=table2.column1
```

> **Note**
>
> 這裡顯示的語法採用一個聯結資料欄配對說明簡單的 equijoin 案例。不過，其他類型的比較條件和多個聯結資料欄配對同樣有效。

例如，下列 WHERE 子句會定義兩個資料欄配對之間的外部聯結。(+) 運算子在這兩個條件中必須連接至相同資料表：

```
where table1.col1 > table2.col1(+)
and table1.col2 = table2.col2(+)
```

使用須知

盡可能使用標準 FROM 子句 OUTER JOIN 語法，而不要在 WHERE 子句中使用 (+) 運算子。包含 (+) 運算子的查詢受到下列規則限制：

- 您只能在 WHERE 子句中使用 (+) 運算子，而且只能參考資料表或檢視的資料欄。
- 您無法將 (+) 運算子套用至表達式。不過，表達式可包含使用 (+) 運算子的資料欄。例如，下列聯結條件會傳回語法錯誤：

  ```
  event.eventid*10(+)=category.catid
  ```

  不過，下列聯結條件有效：

  ```
  event.eventid(+)*10=category.catid
  ```

- 您無法在同時包含 FROM 子句聯結語法的查詢區塊中使用 (+) 運算子。
- 若兩個資料表是透過多個聯結條件聯結，您必須在所有條件中使用 (+) 運算子，或完全不使用。採用混合語法樣式的聯結會做為內部聯結執行，但不會產生警告。
- 若您將外部查詢中的資料表與產生自內部查詢的資料表聯結，則 (+) 運算子不會產生外部聯結。
- 若要使用 (+) 運算子將資料表與其本身進行外部聯結，您必須在 FROM 子句中定義資料表別名，並且在聯結條件中參考這些別名：

  ```
  select count(*)
  from event a, event b
  where a.eventid(+)=b.catid;
  ```

```
count
-------
8798
(1 row)
```

- 您無法將包含 (+) 運算子的聯結條件與 OR 條件或 IN 條件結合。例如：

```
select count(*) from sales, listing
where sales.listid(+)=listing.listid or sales.salesid=0;
ERROR:  Outer join operator (+) not allowed in operand of OR or IN.
```

- 在與兩個以上資料表進行外部聯結的 WHERE 子句中，(+) 運算子只能對特定資料表套用一次。在以下範例中，SALES 資料表無法在兩個連續聯結中使用 (+) 運算子參考。

```
select count(*) from sales, listing, event
where sales.listid(+)=listing.listid and sales.dateid(+)=date.dateid;
ERROR:  A table may be outer joined to at most one other table.
```

- 若 WHERE 子句的外部聯結條件會將 TABLE2 的資料欄與常數進行比較，則將 (+) 運算子套用至資料欄。若您未包含運算子，就會消除 TABLE1 中的外部聯結資料列 (當中限制的資料欄會包含 null)。請參閱下方範例一節。

範例

以下聯結查詢會在 LISTID 資料欄上指定 SALES 和 LISTING 資料表的左外部聯結：

```
select count(*)
from sales, listing
where sales.listid = listing.listid(+);

count
--------
172456
(1 row)
```

下列對等查詢會產生相同的結果，但使用 FROM 子句聯結語法：

```
select count(*)
from sales left outer join listing on sales.listid = listing.listid;

count
```

```
--------
172456
(1 row)
```

SALES 資料表未包含 LISTING 資料表中所有清單的記錄，因為並非所有清單都產生銷售。以下查詢會將 SALES 和 LISTING 進行外部聯結，並從 LISTING 傳回資料列，即使 SALES 資料表回報特定清單 ID 沒有銷售。PRICE 和 COMM 資料欄衍生自 SALES 資料表，其結果集中的不相符資料列會包含 null。

```
select listing.listid, sum(pricepaid) as price,
sum(commission) as comm
from listing, sales
where sales.listid(+) = listing.listid and listing.listid between 1 and 5
group by 1 order by 1;

listid | price  |  comm
--------+--------+--------
1 | 728.00 | 109.20
2 |        |
3 |        |
4 |  76.00 |  11.40
5 | 525.00 |  78.75
(5 rows)
```

請注意，使用 WHERE 子句聯結運算子時，資料表在 FROM 子句中的順序並不重要。

WHERE 子句中較複雜的外部聯結條件範例，就是條件由兩個資料表資料欄之間的比較，以及與常數的比較所構成：

```
where category.catid=event.catid(+) and eventid(+)=796;
```

請注意，(+) 運算子會在兩處使用：一處是資料表之間的對等比較中，另一處是 EVENTID 資料欄的比較條件中。此語法的結果是在評估 EVENTID 的限制時，保留外部聯結資料列。若您從 EVENTID 限制中移除 (+) 運算子，則查詢會將此限制視為篩選條件，而非外部聯結條件的一部分。接著就會從結果集中消除 EVENTID 包含 null 的外部聯結資料列。

以下是說明此行為的完整查詢：

```
select catname, catgroup, eventid
from category, event
where category.catid=event.catid(+) and eventid(+)=796;
```

```
catname | catgroup | eventid
-----------+----------+---------
Classical | Concerts |
Jazz | Concerts |
MLB | Sports   |
MLS | Sports   |
Musicals | Shows    | 796
NBA | Sports   |
NFL | Sports   |
NHL | Sports   |
Opera | Shows    |
Plays | Shows    |
Pop | Concerts |
(11 rows)
```

使用 FROM 子句語法的對等查詢如下所示：

```
select catname, catgroup, eventid
from category left join event
on category.catid=event.catid and eventid=796;
```

若您從此查詢的 WHERE 子句版本中移除第二個 (+) 運算子，則只會傳回 1 個資料列 (eventid=796 的資料列)。

```
select catname, catgroup, eventid
from category, event
where category.catid=event.catid(+) and eventid=796;

catname | catgroup | eventid
-----------+----------+---------
Musicals | Shows    | 796
(1 row)
```

GROUP BY 子句

GROUP BY 子句會識別查詢的分組資料欄。分組資料欄必須在查詢使用標準函數運算彙整時宣告，像是 SUM、AVG 和 COUNT。如需詳細資訊，請參閱 彙總函數。

語法

```
GROUP BY group_by_clause [, ...]
```

```
group_by_clause := {
    expr |
    GROUPING SETS ( () | group_by_clause [, ...] ) |
    ROLLUP ( expr [, ...] ) |
    CUBE ( expr [, ...] )
    }
```

參數

expr

在查詢的選取清單中，資料欄或表達式的清單必須符合非彙整表達式的清單。例如，請考量以下簡單查詢。

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by listid, eventid
order by 3, 4, 2, 1
limit 5;

listid | eventid | revenue | numtix
-------+---------+---------+--------
89397  |      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

在此查詢中，選取清單是由兩個彙整表達式所構成。第一個使用 SUM 函數，第二個使用 COUNT 函數。其餘兩個資料欄 LISTID 和 EVENTID 必須宣告為分組資料欄。

GROUP BY 子句中的表達式也可以使用序數來參考選取清單。例如，前一個範例可縮減如下。

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by 1,2
order by 3, 4, 2, 1
limit 5;
```

```
listid | eventid | revenue | numtix
-------+---------+---------+--------
89397  |      47 |   20.00 |      1
106590 |      76 |   20.00 |      1
124683 |     393 |   20.00 |      1
103037 |     403 |   20.00 |      1
147685 |     429 |   20.00 |      1
(5 rows)
```

GROUPING SETS/ROLLUP/CUBE

您可以使用彙總延伸項目 GROUPING SETS、ROLLUP 和 CUBE，在單一陳述式中執行多個 GROUP BY 操作的工作。如需彙總延伸項目及相關函數的相關資訊，請參閱 彙總延伸項目。

彙總延伸項目

Amazon Redshift 支援彙總延伸項目，可在單一陳述式中執行多個 GROUP BY 操作的工作。

彙總延伸項目的範例會使用 orders 資料表，該資料表會保留電子公司的銷售資料。您可以執行下列操作來建立 orders：

```
CREATE TABLE ORDERS (
    ID INT,
    PRODUCT CHAR(20),
    CATEGORY CHAR(20),
    PRE_OWNED CHAR(1),
    COST DECIMAL
);

INSERT INTO ORDERS VALUES
    (0, 'laptop',       'computers',    'T', 1000),
    (1, 'smartphone',   'cellphones',   'T', 800),
    (2, 'smartphone',   'cellphones',   'T', 810),
    (3, 'laptop',       'computers',    'F', 1050),
    (4, 'mouse',        'computers',    'F', 50);
```

GROUPING SETS

在單一陳述式中計算一或多個群組集。群組集是單一 GROUP BY 子句的集合，也就是一組 0 個或多個資料行，您可以以此將查詢的結果集分組。GROUP BY GROUPING SETS 等同於在由不同資料欄

分組的一個結果集上執行 UNION ALL 查詢。例如，GROUP BY GROUPING SETS((a), (b)) 等同於 GROUP BY a UNION ALL GROUP BY b。

下列範例會傳回訂單資料表的產品根據產品類別和銷售產品種類進行分組的成本。

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY GROUPING SETS(category, product);

       category       |       product        | total
----------------------+----------------------+-------
 computers            |                      |  2100
 cellphones           |                      |  1610
                      | laptop               |  2050
                      | smartphone           |  1610
                      | mouse                |    50

(5 rows)
```

ROLLUP

假設有一個階層，其中之前的資料欄被視為後續資料欄的父項。ROLLUP 會依提供的資料欄將資料分組，並傳回額外的小計資料列 (代表所有分組資料欄層級的總計)，以及已分組的資料列。例如，您可以使用 GROUP BY ROLLUP((a), (b)) 傳回一個先按 a 分組，然後按 b 分組的結果集 (假設 b 是 a 的子區段)。ROLLUP 也會傳回包含整個結果集的資料列，而不會將資料欄分組。

GROUP BY ROLLUP((a), (b)) 等於 GROUP BY GROUPING SETS((a,b), (a), ())。

下列範例會傳回訂單資料表產品先依類別分組，然後依產品分組的成本，其中以產品做為類別的細項。

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY ROLLUP(category, product) ORDER BY 1,2;

       category       |       product        | total
----------------------+----------------------+-------
 cellphones           | smartphone           |  1610
 cellphones           |                      |  1610
 computers            | laptop               |  2050
 computers            | mouse                |    50
 computers            |                      |  2100
                      |                      |  3710
```

```
(6 rows)
```

CUBE

依提供的資料欄將資料分組，並傳回額外的小計資料列 (代表所有分組資料欄層級的總計)，以及已分組的資料列。CUBE 會傳回與 ROLLUP 相同的資料列，同時針對 ROLUP 未涵蓋的每個分組資料欄組合新增額外的小計資料列。例如，您可以使用 GROUP BY CUBE ((a), (b)) 傳回一個先按 a 分組，然後按 b 分組的結果集 (假設 b 是 a 的子區段)，然後再獨自按 b 分組。CUBE 也會傳回包含整個結果集的資料列，而不會將資料欄分組。

GROUP BY CUBE((a), (b)) 等於 GROUP BY GROUPING SETS((a, b), (a), (b), ())。

下列範例會傳回訂單資料表產品先依類別分組，然後依產品分組的成本，其中以產品做為類別的細項。與前面的 ROLLUP 範例不同，陳述式會傳回每個分組資料欄組合的結果。

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY CUBE(category, product) ORDER BY 1,2;

       category       |       product        | total
----------------------+----------------------+-------
 cellphones           | smartphone           |  1610
 cellphones           |                      |  1610
 computers            | laptop               |  2050
 computers            | mouse                |    50
 computers            |                      |  2100
                      | laptop               |  2050
                      | mouse                |    50
                      | smartphone           |  1610
                      |                      |  3710
(9 rows)
```

GROUPING/GROUPING_ID 函數

ROLLUP 和 CUBE 會將 NULL 值新增至結果集，以指出小計資料列。例如，GROUP BY ROLLUP((a), (b)) 會傳回 b 分組資料欄中值為 NULL 的一或多個資料列，以指出這些資料列是分組資料欄中欄位的小計。這些 NULL 值僅用於滿足傳回元組的格式。

當您在儲存 NULL 值本身的關係上使用 ROLLUP 和 CUBE 執行 GROUP BY 操作時，這可能會產生資料列看起來具有相同分組資料欄的結果集。返回上一個範例，如果 b 分組資料欄包含一個儲存的 NULL 值，GROUP BY ROLLUP((a), (b)) 會在 b 分組資料欄中 (不是小計) 傳回值為 NULL 的資料列。

若要區分 ROLLUP 和 CUBE 建立的 NULL 值，以及儲存在資料表本身的 NULL 值，您可以使用 GROUPING 函數或其別名 GROUPING_ID。GROBING 採用單個分組集作為其引數，並且對於結果集中的每一個資料列傳回對應於該位置中分組資料欄的 0 或 1 位元值，然後將該值轉換為整數。如果該位置中的值是由彙總延伸項目建立的 NULL 值，則 GROUPING 會傳回 1。對於所有其他值，包括儲存的 NULL 值，則傳回 0。

例如，GROUPING(category, product) 可以針對指定的資料列傳回下列值，視該資料列的分組資料欄值而定。就此範例而言，資料表中的所有 NULL 值都是彙總延伸項目所建立的 NULL 值。

類別資料欄	產品資料欄	GROUPING 函數位元值	十進制值
非 NULL	非 NULL	00	0
非 NULL	NULL	01	1
NULL	非 NULL	10	2
NULL	NULL	11	3

GROUPING 函數會以下列格式顯示在查詢的 SELECT 清單部分中。

```
SELECT ... [GROUPING( expr )...] ...
  GROUP BY ... {CUBE | ROLLUP| GROUPING SETS} ( expr ) ...
```

下列範例與前面的 CUBE 範例相同，但為其分組集增加了 GROUPING 函數。

```
SELECT category, product,
       GROUPING(category) as grouping0,
       GROUPING(product) as grouping1,
       GROUPING(category, product) as grouping2,
       sum(cost) as total
FROM orders
GROUP BY CUBE(category, product) ORDER BY 3,1,2;

       category       |       product        | grouping0 | grouping1 | grouping2 |
 total
----------------------+----------------------+-----------+-----------+-----------
+-------
```

```
 cellphones           | smartphone           |         0 |         0 |         0 |
 1610
 cellphones           |                      |         0 |         1 |         1 |
 1610
 computers            | laptop               |         0 |         0 |         0 |
 2050
 computers            | mouse                |         0 |         0 |         0 |
 50
 computers            |                      |         0 |         1 |         1 |
 2100
                      | laptop               |         1 |         0 |         2 |
 2050
                      | mouse                |         1 |         0 |         2 |
 50
                      | smartphone           |         1 |         0 |         2 |
 1610
                      |                      |         1 |         1 |         3 |
 3710
(9 rows)
```

部份 ROLLUP 與 CUBE

您只能使用小計的一部分來執行 ROLLUP 和 CUBE 操作。

部分 ROLLUP 和 CUBE 操作的語法如下。

```
GROUP BY expr1, { ROLLUP | CUBE }(expr2, [, ...])
```

在這裡，GROUP BY 子句僅在表達式 expr2 和以後的層級上建立小計資料列。

下列範例會顯示訂單資料表上的部份 ROLLUP 和 CUBE 操作，並先依產品是否為二手產品分組，然後在類別與產品資料欄上執行 ROLLUP 和 CUBE。

```
SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY pre_owned, ROLLUP(category, product) ORDER BY 4,1,2,3;

 pre_owned |       category       |       product        | group_id | total
-----------+----------------------+----------------------+----------+-------
 F         | computers            | laptop               |        0 |  1050
 F         | computers            | mouse                |        0 |    50
```

```
 T         | cellphones           | smartphone           |        0 |  1610
 T         | computers            | laptop               |        0 |  1000
 F         | computers            |                      |        2 |  1100
 T         | cellphones           |                      |        2 |  1610
 T         | computers            |                      |        2 |  1000
 F         |                      |                      |        6 |  1100
 T         |                      |                      |        6 |  2610
(9 rows)

SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY pre_owned, CUBE(category, product) ORDER BY 4,1,2,3;

 pre_owned |       category       |       product        | group_id | total
-----------+----------------------+----------------------+----------+-------
 F         | computers            | laptop               |        0 |  1050
 F         | computers            | mouse                |        0 |    50
 T         | cellphones           | smartphone           |        0 |  1610
 T         | computers            | laptop               |        0 |  1000
 F         | computers            |                      |        2 |  1100
 T         | cellphones           |                      |        2 |  1610
 T         | computers            |                      |        2 |  1000
 F         |                      | laptop               |        4 |  1050
 F         |                      | mouse                |        4 |    50
 T         |                      | laptop               |        4 |  1000
 T         |                      | smartphone           |        4 |  1610
 F         |                      |                      |        6 |  1100
 T         |                      |                      |        6 |  2610
(13 rows)
```

由於二手資料欄不包含在 ROLLUP 和 CUBE 作業中，因此沒有包含所有其他資料列的總計資料列。

串連分組

您可以串連多個 GROUPING SETS/ROLLUP/CUBE 子句，以計算不同層次的小計。串連分組會傳回所提供分組集的笛卡爾乘積。

串連 GROUPING SETS/ROLLUP/CUBE 子句的語法如下。

```
GROUP BY {ROLLUP|CUBE|GROUPING SETS}(expr1[, ...]),
         {ROLLUP|CUBE|GROUPING SETS}(expr1[, ...])[, ...]
```

請考慮下列範例，看看小型串連分組如何產生大型的最終結果集。

```
SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY CUBE(category, product), GROUPING SETS(pre_owned, ())
ORDER BY 4,1,2,3;

 pre_owned |       category       |       product        | group_id | total
-----------+----------------------+----------------------+----------+-------
 F         | computers            | laptop               |        0 |  1050
 F         | computers            | mouse                |        0 |    50
 T         | cellphones           | smartphone           |        0 |  1610
 T         | computers            | laptop               |        0 |  1000
           | cellphones           | smartphone           |        1 |  1610
           | computers            | laptop               |        1 |  2050
           | computers            | mouse                |        1 |    50
 F         | computers            |                      |        2 |  1100
 T         | cellphones           |                      |        2 |  1610
 T         | computers            |                      |        2 |  1000
           | cellphones           |                      |        3 |  1610
           | computers            |                      |        3 |  2100
 F         |                      | laptop               |        4 |  1050
 F         |                      | mouse                |        4 |    50
 T         |                      | laptop               |        4 |  1000
 T         |                      | smartphone           |        4 |  1610
           |                      | laptop               |        5 |  2050
           |                      | mouse                |        5 |    50
           |                      | smartphone           |        5 |  1610
 F         |                      |                      |        6 |  1100
 T         |                      |                      |        6 |  2610
           |                      |                      |        7 |  3710
(22 rows)
```

巢狀分組

您可以使用 GROUPING SETS/ROLLUP/CUBE 操作作為您的 GROUPING SETS expr，形成一個巢狀分組。巢狀 GROUPING SETS 中的子分組會被扁平化。

巢狀分組的語法如下。

```
GROUP BY GROUPING SETS({ROLLUP|CUBE|GROUPING SETS}(expr[, ...])[, ...])
```

請考量下列範例。

```
SELECT category, product, pre_owned,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY GROUPING SETS(ROLLUP(category), CUBE(product, pre_owned))
ORDER BY 4,1,2,3;

       category       |       product        | pre_owned | group_id | total
----------------------+----------------------+-----------+----------+-------
 cellphones           |                      |           |        3 |  1610
 computers            |                      |           |        3 |  2100
                      | laptop               | F         |        4 |  1050
                      | laptop               | T         |        4 |  1000
                      | mouse                | F         |        4 |    50
                      | smartphone           | T         |        4 |  1610
                      | laptop               |           |        5 |  2050
                      | mouse                |           |        5 |    50
                      | smartphone           |           |        5 |  1610
                      |                      | F         |        6 |  1100
                      |                      | T         |        6 |  2610
                      |                      |           |        7 |  3710
                      |                      |           |        7 |  3710
(13 rows)
```

請注意，由於 ROLLUP(category) 和 CUBE(product, pre_owned) 都包含分組集 ()，因此代表總計的資料列將重複。

使用須知

- GROUP BY 子句最多支援 64 個分組集。如果是 ROLLUP 和 CUBE，或是 GROUPING SETS、ROLLUP 和 CUBE 的某些組合，則此限制會套用至隱含的分組集數目。例如，GROUP BY CUBE((a), (b)) 計為 4 個分組集，而不是 2 個。
- 使用彙總延伸項目時，您無法使用常數作為分組資料欄。
- 您無法建立包含重複資料欄的分組集。

HAVING 子句

HAVING 子句會將條件套用至查詢傳回的中繼分組結果集。

語法

```
[ HAVING condition ]
```

例如，您可以限制 SUM 函數的結果：

```
having sum(pricepaid) >10000
```

HAVING 會在套用所有 WHERE 子句條件且完成 GROUP BY 操作之後套用。

條件本身會採用與任何 WHERE 子句條件相同的形式。

使用須知

- HAVING 子句條件中參考的任何資料欄必須是分組資料欄，或是參考彙整函數結果的資料欄。
- 在 HAVING 子句中，您無法指定：
  - 參考選取清單項目的序數。只有 GROUP BY 和 ORDER BY 子句接受序數。

範例

以下查詢會依名稱計算售票總金額，然後消除總金額低於 800,000 USD 的活動。HAVING 條件會套用至選取清單中彙整函數的結果：sum(pricepaid)。

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
group by 1
having sum(pricepaid) > 800000
order by 2 desc, 1;

eventname        |    sum
-----------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
```

以下查詢會計算類似的結果集。不過，在此情況下，HAVING 條件會套用至選取清單中未指定的彙整：sum(qtysold)。未銷售超過 2,000 張票的活動將從最終結果中消除。

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
group by 1
having sum(qtysold) >2000
order by 2 desc, 1;

eventname        |    sum
-----------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
Chicago          |  790993.00
Spamalot         |  714307.00
```

以下查詢會依名稱計算售票總金額，然後消除總金額低於 800,000 USD 的活動。HAVING 條件會套用至使用別名 pp for 的選取清單中彙總函式的結果`sum(pricepaid)`。

```
select eventname, sum(pricepaid) as pp
from sales join event on sales.eventid = event.eventid
group by 1
having pp > 800000
order by 2 desc, 1;

eventname        |    pp
-----------------+-----------
Mamma Mia!       | 1135454.00
Spring Awakening |  972855.00
The Country Girl |  910563.00
Macbeth          |  862580.00
Jersey Boys      |  811877.00
Legally Blonde   |  804583.00
```

QUALIFY 子句

QUALIFY 子句會根據使用者指定的搜尋條件，篩選先前計算的視窗函數結果。您可以使用子句將篩選條件套用至視窗函數的結果，而不需要使用子查詢。

其類似於 HAVING 子句，該子句適用於從 WHERE 子句進一步篩選資料列的條件。QUALIFY 和 HAVING 之間的區別在於，從 QUALIFY 子句篩選的結果可以基於在資料上執行窗口函數的結果。您可以在一個查詢中同時使用 QUALIFY 和 HAVING 子句。

語法

```
QUALIFY condition
```

> **Note**
>
> 如果您直接在 FROM 子句之後使用 QUALIFY 子句，則 FROM 關係名稱必須在 QUALIFY 子句之前指定別名。

範例

本節中的範例使用以下範例資料。

```
create table store_sales (ss_sold_date date, ss_sold_time time,
               ss_item text, ss_sales_price float);
insert into store_sales values ('2022-01-01', '09:00:00', 'Product 1', 100.0),
                               ('2022-01-01', '11:00:00', 'Product 2', 500.0),
                               ('2022-01-01', '15:00:00', 'Product 3', 20.0),
                               ('2022-01-01', '17:00:00', 'Product 4', 1000.0),
                               ('2022-01-01', '18:00:00', 'Product 5', 30.0),
                               ('2022-01-02', '10:00:00', 'Product 6', 5000.0),
                               ('2022-01-02', '16:00:00', 'Product 7', 5.0);
```

下列範例示範如何找到每天 12:00 之後售出的兩個最昂貴的物品。

```
SELECT *
FROM store_sales ss
WHERE ss_sold_time > time '12:00:00'
QUALIFY row_number()
OVER (PARTITION BY ss_sold_date ORDER BY ss_sales_price DESC) <= 2


 ss_sold_date | ss_sold_time |  ss_item  | ss_sales_price
--------------+--------------+-----------+----------------
 2022-01-01   | 17:00:00     | Product 4 |           1000
 2022-01-01   | 18:00:00     | Product 5 |             30
```

```
 2022-01-02   | 16:00:00     | Product 7 |              5
```

然後，您可以找到每天售出的最後一件物品。

```
SELECT *
FROM store_sales ss
QUALIFY last_value(ss_item)
OVER (PARTITION BY ss_sold_date ORDER BY ss_sold_time ASC
      ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) = ss_item;

ss_sold_date | ss_sold_time |  ss_item  | ss_sales_price
-------------+--------------+-----------+----------------
 2022-01-01   | 18:00:00     | Product 5 |             30
 2022-01-02   | 16:00:00     | Product 7 |              5
```

下列範例會傳回與上一個查詢相同的記錄，也就是每天售出的最後一個項目，但不使用 QUERITION 子句。

```
SELECT * FROM (
  SELECT *,
  last_value(ss_item)
  OVER (PARTITION BY ss_sold_date ORDER BY ss_sold_time ASC
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) ss_last_item
  FROM store_sales ss
)
WHERE ss_last_item = ss_item;

 ss_sold_date | ss_sold_time |  ss_item  | ss_sales_price | ss_last_item
-------------+--------------+-----------+----------------+--------------
 2022-01-02   | 16:00:00     | Product 7 |              5 | Product 7
 2022-01-01   | 18:00:00     | Product 5 |             30 | Product 5
```

UNION、INTERSECT 和 EXCEPT

主題

- 語法
- 參數
- 集合運算子的評估順序
- 使用須知
- 範例 UNION 查詢

- 範例 UNION ALL 查詢
- 範例 INTERSECT 查詢
- 範例 EXCEPT 查詢

UNION、INTERSECT 和 EXCEPT 集合運算子可用來比較和合併兩種不同查詢表達式的結果。例如，如果您想知道哪些網站使用者同時是買方和賣家，但其使用者名稱儲存在不同的資料欄或資料表中，您可以找出這兩種類型使用者的交集。如果您想知道哪些網站使用者是買方，但不是賣家，您可以使用 EXCEPT 運算子找出兩份使用者清單之間的差異。如果您想要建構所有使用者的清單，但不考慮角色，您可以使用 UNION 運算子。

語法

```
query
{ UNION [ ALL ] | INTERSECT | EXCEPT | MINUS }
query
```

參數

query

此查詢表達式會以其選取清單形式，對應至接在 UNION、INTERSECT 或 EXCEPT 運算子後面的另一個查詢表達式。兩個表達式必須包含採用相容資料類型的相同輸出資料欄數，否則就無法比較和合併這兩個結果集。集合操作不允許在不同類別的資料類型之間進行隱含轉換；如需詳細資訊，請參閱 類型相容性與轉換。

您可以建構包含無限查詢表達式數目的查詢，並將它們與 UNION、INTERSECT 和 EXCEPT 運算子的任意組合連結。例如，假設資料表 T1、T2 和 T3 包含相容的資料欄集，則以下查詢結構有效：

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

UNION

此集合操作會從兩個查詢表達式傳回資料列，無論資料列衍生自其中一個或兩個表達式。

INTERSECT

此集合操作會傳回衍生自兩個查詢表達式的資料列。未由兩個表達式傳回的資料列則會遭到捨棄。

EXCEPT | MINUS

此集合操作會傳回衍生自兩個查詢表達式之一的資料列。若要限定結果，資料列必須存在第一個結果資料表中，但不能存在第二個資料表中。MINUS 和 EXCEPT 是一模一樣的同義詞。

ALL

ALL 關鍵字會保留 UNION 所產生的任何重複資料列。未使用 ALL 關鍵字時的預設行為是捨棄這些重複項目。不支援 INTERSECT ALL、EXCEPT ALL 和 MINUS ALL。

集合運算子的評估順序

UNION 和 EXCEPT 集合運算子為左關聯。若未指定括號來影響優先順序，則會從左到右評估這些集合運算子的組合。例如，在下列查詢中，T1 和 T2 的 UNION 會先評估，然後在 UNION 結果上執行 EXCEPT 操作：

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

在相同查詢中使用運算子組合時，INTERSECT 運算子的優先順序高於 UNION 和 EXCEPT 運算子。例如，下列查詢會先評估 T2 和 T3 的交集，再將結果與 T1 進行聯集：

```
select * from t1
union
select * from t2
intersect
select * from t3
order by c1;
```

加入括號就可以強制執行不同的評估順序。在下列案例中，T1 和 T2 的聯集結果會與 T3 交集，而查詢可能會產生不同的結果。

```
(select * from t1
union
```

```
select * from t2)
intersect
(select * from t3)
order by c1;
```

使用須知

- 集合操作查詢的結果中傳回的資料欄名稱，是來自第一個查詢表達式的資料表中的資料欄名稱 (或別名)。這些資料欄名稱可能會造成誤導，因為資料欄中的值是從任一邊集合運算子的資料表衍生，所以建議您為結果集提供有意義的別名。
- 先於集合運算子的查詢表達式不應包含 ORDER BY 子句。只有在包含集合運算子的查詢結尾使用 ORDER BY 子句時，該子句才會產生有意義的排序結果。在此情況下，ORDER BY 子句會套用至所有集合操作的最終結果。最外層的查詢也可包含標準 LIMIT 和 OFFSET 子句。
- 當集合運算子查詢傳回小數結果時，對應的結果資料欄就會提升，以傳回相同的精確度和小數位數。例如，在以下查詢中，T1.REVENUE 是 DECIMAL(10,2) 資料欄，而 T2.REVENUE 是 DECIMAL(8,4) 資料欄，小數結果會提升為 DECIMAL(12,4)：

  ```
  select t1.revenue union select t2.revenue;
  ```

  小數位數為 4，因為這是兩個資料欄的小數位數上限。精確度為 12，因為 T1.REVENUE 要求小數點左邊有 8 位數 (12 - 4 = 8)。此類型提升可確保 UNION 兩邊的所有值都能納入結果中。若是 64 位元值，最高結果精確度為 19，而結果小數位數上限為 18。若是 128 位元值，最高結果精確度為 38，而結果小數位數上限為 37。

  若產生的資料類型超過 Amazon Redshift 精確度和小數位數限制，查詢就會傳回錯誤。
- 在集合操作中，若每個對應資料欄配對的這兩個資料值為等於或兩者皆為 NULL，則這兩個資料列會視為相同。例如，若資料表 T1 和 T2 都包含一個資料欄和一個資料列，而該資料列在兩個資料表中都是 NULL，則對這些資料表執行 INTERSECT 操作就會傳回該資料列。

範例 UNION 查詢

在下列 UNION 查詢中，SALES 資料表中的資料列會與 LISTING 資料表中的資料列合併。會從每個資料表選取三個相容的資料欄；在此情況下，對應的資料欄會有相同的名稱和資料類型。

最終結果集是依 LISTING 資料表中的第一欄排序，且限於擁有最高 LISTID 值的 5 個資料列。

```
select listid, sellerid, eventid from listing
union select listid, sellerid, eventid from sales
```

```
order by listid, sellerid, eventid desc limit 5;

listid | sellerid | eventid
--------+----------+---------
1 |    36861 |    7872
2 |    16002 |    4806
3 |    21461 |    4256
4 |     8117 |    4337
5 |     1616 |    8647
(5 rows)
```

以下範例說明如何將常值新增至 UNION 查詢的輸出，以便查看結果集中的每個資料列是由哪個查詢表達式所產生。查詢會將來自第一個查詢表達式的資料列識別為 "B" (表示買方)，來自第二個查詢表達式的資料列識別為 "S" (表示賣家)。

查詢會識別價值 10,000 USD 以上的票券交易買方和賣家。UNION 運算子任一邊的兩個查詢表達式之間唯一的差異，就是 SALES 資料表的聯結資料欄。

```
select listid, lastname, firstname, username,
pricepaid as price, 'S' as buyorsell
from sales, users
where sales.sellerid=users.userid
and pricepaid >=10000
union
select listid, lastname, firstname, username, pricepaid,
'B' as buyorsell
from sales, users
where sales.buyerid=users.userid
and pricepaid >=10000
order by 1, 2, 3, 4, 5;

listid | lastname | firstname | username |   price   | buyorsell
--------+----------+-----------+----------+-----------+-----------
209658 | Lamb     | Colette   | VOR15LYI |  10000.00 | B
209658 | West     | Kato      | ELU81XAA |  10000.00 | S
212395 | Greer    | Harlan    | GXO71KOC |  12624.00 | S
212395 | Perry    | Cora      | YWR73YNZ |  12624.00 | B
215156 | Banks    | Patrick   | ZNQ69CLT |  10000.00 | S
215156 | Hayden   | Malachi   | BBG56AKU |  10000.00 | B
(6 rows)
```

以下範例使用 UNION ALL 運算子，因為重複的資料列 (若找到) 需保留在結果中。若是特定活動 ID 系列，查詢會針對與各個活動相關聯的每筆銷售傳回 0 或更多資料列，並針對該活動的每份清單傳回 0 或 1。LISTING 和 EVENT 資料表中每個資料列的活動 ID 都是唯一的，但 SALES 資料表中相同的活動和清單 ID 組合可能會有多筆銷售。

結果集中的第三個資料欄會識別資料列的來源。若是來自 SALES 資料表，則會在 SALESROW 資料欄中標示為「Yes (是)」 (SALESROW 是 SALES.LISTID 的別名)。若資料列來自 LISTING 資料表，則會在 SALESROW 資料欄中標示為「No (否)」。

在此情況下，結果集會包含活動 7787、清單 500 的三個銷售資料列。換句話說，此清單與活動組合發生了三筆不同的交易。另外兩份清單 501 和 502 並未產生任何銷售，因此查詢針對這些清單 ID 產生的唯一資料列是來自 LISTING 資料表 (SALESROW = 'No')。

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
7787 |    500 | Yes
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(6 rows)
```

若您執行相同查詢，但未使用 ALL 關鍵字，結果只會保留其中一項銷售交易。

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;
```

```
eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(4 rows)
```

範例 UNION ALL 查詢

以下範例使用 UNION ALL 運算子，因為重複的資料列 (若找到) 需保留在結果中。若是特定活動 ID 系列，查詢會針對與各個活動相關聯的每筆銷售傳回 0 或更多資料列，並針對該活動的每份清單傳回 0 或 1。LISTING 和 EVENT 資料表中每個資料列的活動 ID 都是唯一的，但 SALES 資料表中相同的活動和清單 ID 組合可能會有多筆銷售。

結果集中的第三個資料欄會識別資料列的來源。若是來自 SALES 資料表，則會在 SALESROW 資料欄中標示為「Yes (是)」 (SALESROW 是 SALES.LISTID 的別名)。若資料列來自 LISTING 資料表，則會在 SALESROW 資料欄中標示為「No (否)」。

在此情況下，結果集會包含活動 7787、清單 500 的三個銷售資料列。換句話說，此清單與活動組合發生了三筆不同的交易。另外兩份清單 501 和 502 並未產生任何銷售，因此查詢針對這些清單 ID 產生的唯一資料列是來自 LISTING 資料表 (SALESROW = 'No')。

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
7787 |    500 | Yes
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(6 rows)
```

若您執行相同查詢，但未使用 ALL 關鍵字，結果只會保留其中一項銷售交易。

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

eventid | listid | salesrow
---------+--------+----------
7787 |    500 | No
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(4 rows)
```

範例 INTERSECT 查詢

比較下列範例與第一個 UNION 範例。這兩個範例的唯一差異在於使用的集合運算子，但結果非常不同。只有其中一個資料列相同：

```
235494 |    23875 |    8771
```

這是在限制的 5 個資料列結果中，唯一同時存在兩個資料表中的資料列。

```
select listid, sellerid, eventid from listing
intersect
select listid, sellerid, eventid from sales
order by listid desc, sellerid, eventid
limit 5;

listid | sellerid | eventid
--------+----------+---------
235494 |    23875 |    8771
235482 |     1067 |    2667
235479 |     1589 |    7303
235476 |    15550 |     793
235475 |    22306 |    7848
(5 rows)
```

以下查詢會尋找三月份同時於紐約市和洛杉磯的場館舉行的活動 (有售票)。兩個查詢表達式之間唯一的差異就是 VENUECITY 資料欄的限制條件。

```
select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='Los Angeles'
intersect
select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='New York City'
order by eventname asc;

eventname
----------------------------
A Streetcar Named Desire
Dirty Dancing
Electra
Running with Annalise
Hairspray
Mary Poppins
November
Oliver!
Return To Forever
Rhinoceros
South Pacific
The 39 Steps
The Bacchae
The Caucasian Chalk Circle
The Country Girl
Wicked
Woyzeck
(16 rows)
```

範例 EXCEPT 查詢

TICKIT 資料庫中的 CATEGORY 資料表包含以下 11 個資料列：

```
 catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
   1   | Sports   | MLB       | Major League Baseball
   2   | Sports   | NHL       | National Hockey League
   3   | Sports   | NFL       | National Football League
   4   | Sports   | NBA       | National Basketball Association
```

```
   5   | Sports   | MLS       | Major League Soccer
   6   | Shows    | Musicals  | Musical theatre
   7   | Shows    | Plays     | All non-musical theatre
   8   | Shows    | Opera     | All opera and light opera
   9   | Concerts | Pop       | All rock and pop music concerts
  10   | Concerts | Jazz      | All jazz singers and bands
  11   | Concerts | Classical | All symphony, concerto, and choir concerts
(11 rows)
```

假設 CATEGORY_STAGE 資料表 (臨時資料表) 包含一個額外的資料列：

```
 catid | catgroup |  catname  |                 catdesc
-------+----------+-----------+--------------------------------------------
1 | Sports   | MLB       | Major League Baseball
2 | Sports   | NHL       | National Hockey League
3 | Sports   | NFL       | National Football League
4 | Sports   | NBA       | National Basketball Association
5 | Sports   | MLS       | Major League Soccer
6 | Shows    | Musicals  | Musical theatre
7 | Shows    | Plays     | All non-musical theatre
8 | Shows    | Opera     | All opera and light opera
9 | Concerts | Pop       | All rock and pop music concerts
10 | Concerts | Jazz      | All jazz singers and bands
11 | Concerts | Classical | All symphony, concerto, and choir concerts
12 | Concerts | Comedy    | All stand up comedy performances
(12 rows)
```

傳回兩個資料表之間的差異。換句話說，會傳回 CATEGORY_STAGE 資料表而非 CATEGORY 資料表中的資料列：

```
select * from category_stage
except
select * from category;

catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
12 | Concerts | Comedy  | All stand up comedy performances
(1 row)
```

以下對等查詢使用同義詞 MINUS。

```
select * from category_stage
```

```
minus
select * from category;

catid | catgroup | catname |             catdesc
-------+----------+---------+----------------------------------
12 | Concerts | Comedy  | All stand up comedy performances
(1 row)
```

若您將 SELECT 表達式的順序反轉，則查詢不會傳回任何資料列。

ORDER BY 子句

主題

- 語法
- 參數
- 使用須知
- ORDER BY 的範例

ORDER BY 子句會排序查詢的結果集。

語法

```
[ ORDER BY expression [ ASC | DESC ] ]
[ NULLS FIRST | NULLS LAST ]
[ LIMIT { count | ALL } ]
[ OFFSET start ]
```

參數

運算式

此表達式會排序查詢結果集的順序，通常是藉由指定選取清單中的一個或多個資料欄。結果會根據二進位 UTF-8 順序傳回。您還可以指定下列項目：

- 不在選取清單中的資料欄
- 由查詢所參考資料表中的一個或多個資料欄構成的表達式
- 代表選取清單項目位置的序數 (或是，若沒有選取清單的話，則為資料欄在資料表中的位置)
- 定義選取清單項目的別名

當 ORDER BY 子句包含多個表達式時，結果集會根據第一個表達式排序，然後對擁有與第一個表達式相符之值的資料列套用第二個表達式，以此類推。

ASC | DESC

此選項會定義表達式的排序順序，如下所示：

- ASC：遞增 (例如，數值從低到高，字元字串 'A' 到 'Z')。若未指定選項，資料會預設為遞增排序。
- DESC：遞減 (數值從高到低，字串 'Z' 到 'A')。

NULLS FIRST | NULLS LAST

這些選項指定 NULL 值應該排序在最前 (在非 null 值之前) 或排序在最後 (在非 null 值之後)。根據預設，依 ASC 順序排序時，NULL 值排在最後面，而依 DESC 順序排序時，則排在最前面。

LIMIT number | ALL

此選項會控制查詢傳回的排序資料列數。LIMIT 數字必須是正整數；最大值為 2147483647。

LIMIT 0 不會傳回任何資料列。您可以使用此語法進行測試：查看查詢執行情形 (不顯示任何資料列)，或從資料表傳回資料欄清單。若您使用 LIMIT 0 傳回資料欄清單，則 ORDER BY 子句是多餘的。預設值為 LIMIT ALL。

OFFSET start

此選項會指定先略過 start 之前的資料列數，再開始傳回資料列。OFFSET 數字必須是正整數；最大值為 2147483647。搭配 LIMIT 選項使用時，會先略過 OFFSET 資料列，再開始計算傳回的 LIMIT 資料列。如果未使用 LIMIT 選項，則結果集中的資料列數會減掉略過的資料列數。OFFSET 子句略過的資料列仍須經過掃描，因此使用較大的 OFFSET 值可能會導致效率不佳。

使用須知

請注意以下使用 ORDER BY 子句的預期行為：

- NULL 值會視為「高於」所有其他值。使用預設的遞增排序順序時，NULL 值會排列在最後面。若要變更此行為，請使用 NULLS FIRST 選項。
- 若查詢未包含 ORDER BY 子句，系統傳回的結果集當中就不會有可預測的資料列排列順序。執行相同的查詢兩次，可能會傳回依不同順序排列的結果集。

- LIMIT 和 OFFSET 選項可在沒有 ORDER BY 子句的情況下使用；不過，若要傳回一致的資料列集，請使用這些選項搭配 ORDER BY。
- 在任何平行系統中，例如 Amazon Redshift，當 ORDER BY 子句無法產生唯一排列順序時，資料列的順序便會不確定。也就是說，如果 ORDER BY 表達式產生重複的值，則這些資料列的傳回順序可能與其他系統不同，也可能隨著每次執行 Amazon Redshift 而有所不同。
- Amazon Redshift 不支援 ORDER BY 子句中的字符文字。

ORDER BY 的範例

從 CATEGORY 資料表傳回全部 11 列，並依第二個資料欄 CATGROUP 排序。若結果擁有相同的 CATGROUP 值，則依字元字串的長度排列 CATDESC 資料欄的值。然後，依 CATID 和 CATNAME 欄排序。

```
select * from category order by 2, length(catdesc), 1, 3;

catid | catgroup |  catname  |                 catdesc
------+----------+-----------+--------------------------------------
10    | Concerts | Jazz      | All jazz singers and bands
9     | Concerts | Pop       | All rock and pop music concerts
11    | Concerts | Classical | All symphony, concerto, and choir conce
6     | Shows    | Musicals  | Musical theatre
7     | Shows    | Plays     | All non-musical theatre
8     | Shows    | Opera     | All opera and light opera
5     | Sports   | MLS       | Major League Soccer
1     | Sports   | MLB       | Major League Baseball
2     | Sports   | NHL       | National Hockey League
3     | Sports   | NFL       | National Football League
4     | Sports   | NBA       | National Basketball Association
(11 rows)
```

從 SALES 資料表傳回選取的欄，並依最高 QTYSOLD 值排序。將結果限制為前 10 個資料列：

```
select salesid, qtysold, pricepaid, commission, saletime from sales
order by qtysold, pricepaid, commission, salesid, saletime desc
limit 10;

salesid | qtysold | pricepaid | commission |     saletime
--------+---------+-----------+------------+---------------------
15401   |       8 |    272.00 |      40.80 | 2008-03-18 06:54:56
61683   |       8 |    296.00 |      44.40 | 2008-11-26 04:00:23
```

```
90528   |       8 |    328.00 |      49.20 | 2008-06-11 02:38:09
74549   |       8 |    336.00 |      50.40 | 2008-01-19 12:01:21
130232  |       8 |    352.00 |      52.80 | 2008-05-02 05:52:31
55243   |       8 |    384.00 |      57.60 | 2008-07-12 02:19:53
16004   |       8 |    440.00 |      66.00 | 2008-11-04 07:22:31
489     |       8 |    496.00 |      74.40 | 2008-08-03 05:48:55
4197    |       8 |    512.00 |      76.80 | 2008-03-23 11:35:33
16929   |       8 |    568.00 |      85.20 | 2008-12-19 02:59:33
(10 rows)
```

使用 LIMIT 0 語法會傳回一份資料欄清單，但沒有資料列：

```
select * from venue limit 0;
venueid | venuename | venuecity | venuestate | venueseats
---------+-----------+-----------+------------+------------
(0 rows)
```

CONNECT BY 子句

CONNECT BY 子句會指定階層中資料列之間的關係。您可以使用 CONNECT BY，透過將資料表聯結至自身並處理階層式資料，以階層順序選取資料列。例如，您可以使用它來遞迴循環組織結構圖和列出資料。

階層式查詢會以下列順序處理：

1. 如果 FROM 子句具有聯結，則會先處理它。
2. CONNECT BY 子句會受到評估。
3. WHERE 子句會受到評估。

語法

```
[START WITH start_with_conditions]
CONNECT BY connect_by_conditions
```

> **Note**
>
> 雖然 START 和 CONNECT 不是保留字，但如果您在查詢中使用 START 和 CONNECT 做為資料表別名，請使用分隔識別碼 (雙引號) 或 AS，以避免在執行階段失敗。

```
SELECT COUNT(*)
FROM Employee "start"
CONNECT BY PRIOR id = manager_id
START WITH name = 'John'
```

```
SELECT COUNT(*)
FROM Employee AS start
CONNECT BY PRIOR id = manager_id
START WITH name = 'John'
```

參數

start_with_conditions

指定階層根資料列的條件

connect_by_conditions

指定階層父項資料列與子資料列之間關係的條件。至少有一個條件必須使用用來參照父資料列的一元運算子來限定。

```
PRIOR column = expression
-- or
expression > PRIOR column
```

運算子

您可以在 CONNECT BY 查詢中使用以下運算子。

LEVEL

傳回階層中目前資料列層級的虛擬資料欄。針對根資料列傳回 1，針對根資料列的子項傳回 2，依此類推。

PRIOR

一元運算子，用於評估階層中目前資料列之父資料列的運算式。

範例

下列範例是 CONNECT BY 查詢，此查詢會傳回直接或間接向 John 報告的員工人數，不超過 4 個層級。

```
SELECT id, name, manager_id
FROM employee
WHERE LEVEL < 4
START WITH name = 'John'
CONNECT BY PRIOR id = manager_id;
```

以下為查詢結果。

```
id      name      manager_id
------+----------+--------------
  101     John        100
  102     Jorge       101
  103     Kwaku       101
  110     Liu         101
  201     Sofía       102
  106     Mateo       102
  110     Nikki       103
  104     Paulo       103
  105     Richard     103
  120     Saanvi      104
  200     Shirley     104
  205     Zhang       104
```

此範例的資料表定義：

```
CREATE TABLE employee (
   id INT,
   name VARCHAR(20),
   manager_id INT
   );
```

以下是插入到資料表中的資料列。

```
INSERT INTO employee(id, name, manager_id)  VALUES
(100, 'Carlos', null),
(101, 'John', 100),
(102, 'Jorge', 101),
(103, 'Kwaku', 101),
(110, 'Liu', 101),
(106, 'Mateo', 102),
(110, 'Nikki', 103),
(104, 'Paulo', 103),
```

```
(105, 'Richard', 103),
(120, 'Saanvi', 104),
(200, 'Shirley', 104),
(201, 'Sofía', 102),
(205, 'Zhang', 104);
```

以下是 John 的部門組織結構圖。

子查詢範例

下列範例顯示將子查詢納入 SELECT 查詢的不同方式。請參閱 JOIN 範例，了解另一個使用子查詢的範例。

SELECT 清單子查詢

以下範例包含 SELECT 清單中的子查詢。此子查詢為純量：它只會傳回一個資料欄和一個值，該值會在從外部查詢傳回的每個資料列的結果中重複出現。查詢會比較子查詢運算的 Q1SALES 值與 2008 年另兩季 (2 和 3) 的銷售數字，如外部查詢所定義。

```
select qtr, sum(pricepaid) as qtrsales,
(select sum(pricepaid)
from sales join date on sales.dateid=date.dateid
where qtr='1' and year=2008) as q1sales
from sales join date on sales.dateid=date.dateid
where qtr in('2','3') and year=2008
group by qtr
order by qtr;

qtr  |  qtrsales   |   q1sales
-------+-------------+-------------
2     | 30560050.00 | 24742065.00
3     | 31170237.00 | 24742065.00
(2 rows)
```

WHERE 子句子查詢

以下範例包含 WHERE 子句中的資料表子查詢。此子查詢會產生多個資料列。在此情況下，資料列只會包含一個資料欄，但資料表子查詢可包含多個資料欄和資料列，就像任何其他資料表一樣。

查詢會尋找票券銷售量最高的前 10 名賣家。前 10 名清單受到子查詢的限制，會移除居住在有售票場地之城市的使用者。此查詢可透過不同的方式撰寫；例如，子查詢可重寫為主查詢內的聯結。

```
select firstname, lastname, city, max(qtysold) as maxsold
from users join sales on users.userid=sales.sellerid
where users.city not in(select venuecity from venue)
group by firstname, lastname, city
order by maxsold desc, city desc
limit 10;

firstname | lastname  |      city      | maxsold
----------+-----------+----------------+---------
Noah      | Guerrero | Worcester      |       8
Isadora   | Moss     | Winooski       |       8
Kieran    | Harrison | Westminster    |       8
Heidi     | Davis    | Warwick        |       8
Sara      | Anthony  | Waco           |       8
Bree      | Buck     | Valdez         |       8
Evangeline | Sampson  | Trenton        |       8
Kendall   | Keith    | Stillwater     |       8
Bertha    | Bishop   | Stevens Point  |       8
Patricia  | Anderson | South Portland |       8
(10 rows)
```

WITH 子句子查詢

請參閱WITH 子句。

相互關聯子查詢

以下範例包含 WHERE 子句中的相互關聯子查詢；這類子查詢的資料欄與外部查詢產生的資料欄之間包含一項或多項相互關聯。在此情況下，相互關聯為 `where s.listid=l.listid`。針對外部查詢產生的每個資料列，子查詢會執行以限定或取消限定資料列。

```
select salesid, listid, sum(pricepaid) from sales s
where qtysold=
(select max(numtickets) from listing l
where s.listid=l.listid)
group by 1,2
order by 1,2
limit 5;

salesid | listid |   sum
--------+--------+----------
 27     |     28 | 111.00
 81     |    103 | 181.00
```

```
 142    |    149 | 240.00
 146    |    152 | 231.00
 194    |    210 | 144.00
(5 rows)
```

不支援的相互關聯子查詢模式

查詢規劃器會使用一種查詢重寫方法，稱為子查詢解除相互關聯，在 MPP 環境中最佳化數種相互關聯子查詢模式以供執行。有幾種類型的相互關聯子查詢採用了 Amazon Redshift 無法解除相互關聯也不支援的模式。包含下列相互關聯參考的查詢會傳回錯誤：

- 略過查詢區塊的相互關聯參考，也稱為「略過層級相互關聯參考」。例如，在下列查詢中，包含相互關聯參考的區塊和略過的區塊會以 NOT EXISTS 述詞連接：

  ```
  select event.eventname from event
  where not exists
  (select * from listing
  where not exists
  (select * from sales where event.eventid=sales.eventid));
  ```

  在此案例中略過的區塊是對 LISTING 資料表的子查詢。相互關聯參考會將 EVENT 和 SALES 資料表相互關聯。
- 來自子查詢的相互關聯參考，它是外部查詢中 ON 子句的一部分：

  ```
  select * from category
  left join event
  on category.catid=event.catid and eventid =
  (select max(eventid) from sales where sales.eventid=event.eventid);
  ```

  ON 子句包含從子查詢中 SALES 對外部查詢中 EVENT 的相互關聯參考。
- Null 敏感的相互關聯會參考 Amazon Redshift 系統資料表。例如：

  ```
  select attrelid
  from stv_locks sl, pg_attribute
  where sl.table_id=pg_attribute.attrelid and 1 not in
  (select 1 from pg_opclass where sl.lock_owner = opcowner);
  ```
- 來自子查詢內的相互關聯參考，當中包含視窗函數。

  ```
  select listid, qtysold
  ```

```
from sales s
where qtysold not in
(select sum(numtickets) over() from listing l where s.listid=l.listid);
```

- GROUP BY 資料欄中對相互關聯子查詢結果的參考。例如：

```
select listing.listid,
(select count (sales.listid) from sales where sales.listid=listing.listid) as list
from listing
group by list, listing.listid;
```

- 子查詢中使用彙整函數和 GROUP BY 子句的相互關聯參考，會透過 IN 述詞連接至外部查詢。(此限制不會套用至 MIN 和 MAX 彙整函數)。例如：

```
select * from listing where listid in
(select sum(qtysold)
from sales
where numtickets>4
group by salesid);
```

SELECT INTO

選取任何查詢所定義的資料列，然後將它們插入新的資料表中。您可以指定是否建立臨時或永久資料表。

語法

```
[ WITH with_subquery [, ...] ]
SELECT
[ TOP number ] [ ALL | DISTINCT ]
* | expression [ AS output_name ] [, ...]
INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table
[ FROM table_reference [, ...] ]
[ WHERE condition ]
[ GROUP BY expression [, ...] ]
[ HAVING condition [, ...] ]
[ { UNION | INTERSECT | { EXCEPT | MINUS } } [ ALL ] query ]
[ ORDER BY expression
[ ASC | DESC ]
[ LIMIT { number | ALL } ]
[ OFFSET start ]
```

如需有關此命令之參數的詳細資訊，請參閱 SELECT。

範例

從 EVENT 資料表選取所有資料列，並建立 NEWEVENT 資料表：

```
select * into newevent from event;
```

將彙整查詢的結果選入名為 PROFITS 的臨時資料表中：

```
select username, lastname, sum(pricepaid-commission) as profit
into temp table profits
from sales, users
where sales.sellerid=users.userid
group by 1, 2
order by 3 desc;
```

SET

設定伺服器組態參數的值。使用 SET 命令，僅覆寫目前工作階段持續時間的設定。

使用 RESET 命令將參數恢復為其預設值。

您可以採用數種方式來變更伺服器組態參數。如需詳細資訊，請參閱 修改伺服器組態。

語法

```
SET { [ SESSION | LOCAL ]
{ SEED | parameter_name } { TO | = }
{ value | 'value' | DEFAULT } |
SEED TO value }
```

以下陳述式會設定工作階段內容變數的值。

```
SET { [ SESSION | LOCAL ]
variable_name { TO | = }
{ value | 'value'  }
```

參數

SESSION

指出目前工作階段的設定有效。預設值.

variable_name

指定為工作階段設定的內容變數名稱。

命名慣例是由點分隔的兩部分名稱，例如 identifier.identifier。只允許使用一個點分隔符號。使用符合 Amazon Redshift 標準識別碼規則的 identifier，如需詳細資訊，請參閱 名稱與識別碼。不允許使用分隔的識別碼。

LOCAL

指出目前交易的設定有效。

SEED TO value

設定 RANDOM 函數產生亂數時要使用的內部種子。

SET SEED 會採用介於 0 和 1 之間的數值，並將此數字乘以 (2^{31}-1) 以搭配 RANDOM 函數 函數使用。若您在進行多次 RANDOM 呼叫之前使用 SET SEED，RANDOM 就會依可預測的順序產生數字。

parameter_name

要設定的參數名稱。如需參數的詳細資訊，請參閱 修改伺服器組態。

值

新參數值。使用單引號將值設定為特定字串。若使用 SET SEED，此參數會包含 SEED 值。

DEFAULT

將參數設定為預設值。

範例

變更目前工作階段的參數

下列範例會設定 datestyle：

```
set datestyle to 'SQL,DMY';
```

設定查詢群組以進行工作負載管理

若查詢群組在佇列定義中列為叢集 WLM 組態的一部分，您就可以對列出的查詢群組名稱設定 QUERY_GROUP 參數。後續查詢會指派至相關聯的查詢佇列。QUERY_GROUP 設定在工作階段期間仍然有效，或直到遇到 RESET QUERY_GROUP 命令為止。

此範例會在查詢群組 'priority' 中執行兩個查詢，然後重設查詢群組。

```
set query_group to 'priority';
select tbl, count(*)from stv_blocklist;
select query, elapsed, substring from svl_qlog order by query desc limit 5;
reset query_group;
```

如需詳細資訊，請參閱 實作工作負載管理。

變更工作階段的預設識別名稱空間

資料庫使用者可以設定default_identity_namespace。此範例顯示如SET SESSION何使用覆寫目前階段作業期間的設定，然後顯示新的身分識別提供者值。當您搭配 Redshift 和 IAM 身分中心使用身分識別提供者時，最常使用此功能。如需有關透過 Redshift 使用身分識別提供者的詳細資訊，請參閱將 Redshift 與 IAM 身分中心連線，為使用者提供單一登入體驗。

```
SET SESSION default_identity_namespace = 'MYCO';

SHOW default_identity_namespace;
```

執行命令之後，您可以執行 GRANT 陳述式或 CREATE 陳述式，如下所示：

```
GRANT SELECT ON TABLE mytable TO alice;

GRANT UPDATE ON TABLE mytable TO salesrole;

CREATE USER bob password 'md50c983d1a624280812631c5389e60d48c';
```

在這個例子中，設定預設識別名稱空間的效果等同於在每個識別前面加上命名空間。在此範例中，alice會取代為MYCO:alice。如需使用 IAM 身分中心與 Redshift 組態相關之設定的詳細資訊，請參閱ALTER SYSTEM和。ALTER IDENTITY PROVIDER

設定查詢群組的標籤

QUERY_GROUP 參數會為一個或多個在相同工作階段中的 SET 命令後面執行的查詢定義標籤。接著此標籤會在查詢執行時記錄，並且可用來限制從 STL_QUERY 和 STV_INFLIGHT 系統資料表以及 SVL_QLOG 檢視傳回的結果。

```
show query_group;
query_group
-------------
unset
(1 row)

set query_group to '6 p.m.';


show query_group;
query_group
-------------
6 p.m.
(1 row)

select * from sales where salesid=500;
salesid | listid | sellerid | buyerid | eventid | dateid | ...
---------+--------+----------+---------+---------+--------+-----
500 |    504 |     3858 |    2123 |    5871 |   2052 | ...
(1 row)

reset query_group;

select query, trim(label) querygroup, pid, trim(querytxt) sql
from stl_query
where label ='6 p.m.';
query | querygroup |  pid  |                  sql
-------+------------+-------+----------------------------------------
57 | 6 p.m.     | 30711 | select * from sales where salesid=500;
(1 row)
```

查詢群組標籤是實用的機制，方便用來隔離指令碼中執行的個別查詢或查詢群組。您不需要依 ID 識別和追蹤查詢；可依其標籤進行追蹤。

設定產生亂數的 Seed 值

以下範例會使用 SEED 選項搭配 SET，讓 RANDOM 函數依可預測的順序產生數字。

首先傳回三個 RANDOM 整數，但不先設定 SEED 值：

```
select cast (random() * 100 as int);
int4
------
6
(1 row)

select cast (random() * 100 as int);
int4
------
68
(1 row)

select cast (random() * 100 as int);
int4
------
56
(1 row)
```

現在，將 SEED 值設為 .25，並傳回三個以上的 RANDOM 數字：

```
set seed to .25;

select cast (random() * 100 as int);
int4
------
21
(1 row)

select cast (random() * 100 as int);
int4
------
79
(1 row)

select cast (random() * 100 as int);
int4
------
12
(1 row)
```

最後，將 SEED 值重設為 .25，並驗證 RANDOM 是否傳回與前三個呼叫相同的結果：

```
set seed to .25;

select cast (random() * 100 as int);
int4
------
21
(1 row)

select cast (random() * 100 as int);
int4
------
79
(1 row)

select cast (random() * 100 as int);
int4
------
12
(1 row)
```

以下範例會設定自訂內容變數。

```
SET app_context.user_id TO 123;
SET app_context.user_id TO 'sample_variable_value';
```

SET SESSION AUTHORIZATION

設定目前工作階段的使用者名稱。

例如，您可以使用 SET SESSION AUTHORIZATION 命令，以未經授權使用者的身分暫時執行工作階段或交易來測試資料庫存取。您必須是資料庫超級使用者才能執行此命令。

語法

```
SET [ LOCAL ] SESSION AUTHORIZATION { user_name | DEFAULT }
```

參數

LOCAL

指出目前交易的設定有效。省略此參數會指出目前工作階段的設定有效。

user_name

要設定的使用者名稱。使用者名稱可做為識別碼或字串常值寫入。

DEFAULT

將工作階段使用者名稱設為預設值。

範例

以下範例會將目前工作階段的使用者名稱設為 dwuser：

```
SET SESSION AUTHORIZATION 'dwuser';
```

以下範例會將目前交易的使用者名稱設為 dwuser：

```
SET LOCAL SESSION AUTHORIZATION 'dwuser';
```

此範例會將目前工作階段的使用者名稱設為預設使用者名稱：

```
SET SESSION AUTHORIZATION DEFAULT;
```

SET SESSION CHARACTERISTICS

此命令已棄用。

SHOW

顯示伺服器組態參數的目前值。若 SET 命令有效，此值可為目前工作階段專用。如需組態參數的清單，請參閱 組態參考。

語法

```
SHOW { parameter_name | ALL }
```

以下陳述式會顯示工作階段內容變數的目前值。如果該變數不存在，Amazon Redshift 會擲回錯誤。

```
SHOW variable_name
```

參數

parameter_name

顯示指定之參數的目前值。

ALL

顯示所有參數的目前值。

variable_name

顯示指定變數的目前值。

範例

以下範例會顯示 query_group 參數的值：

```
show query_group;

query_group

unset
(1 row)
```

以下範例會顯示所有參數與其值的清單：

```
show all;
name        |   setting
--------------------+--------------
datestyle          | ISO, MDY
extra_float_digits | 0
query_group        | unset
search_path        | $user,public
statement_timeout  | 0
```

下列範例會顯示指定變數的目前值。

```
SHOW app_context.user_id;
```

SHOW COLUMNS

顯示資料表中的資料欄清單，以及某些資料欄屬性。

每個輸出資料列都包含以逗號分隔的資料庫名稱、結構描述名稱、資料表名稱、資料欄名稱、序號位置、資料欄預設值、可為 Null、資料類型、字元最大長度、數值有效位數及備註。如需這些屬性的相關資訊，請參閱 SVV_ALL_COLUMNS。

如果 SHOW COLUMNS 命令所產生的資料欄超過 10,000 個，則會傳回錯誤。

語法

```
SHOW COLUMNS FROM TABLE database_name.schema_name.table_name [LIKE 'filter_pattern']
 [LIMIT row_limit ]
```

參數

database_name

包含所要列出資料表的資料庫名稱。

若要在中顯示表格 AWS Glue Data Catalog，請指定 (awsdatacatalog) 作為資料庫名稱，並確定系統組態data_catalog_auto_mount設定為true。如需詳細資訊，請參閱 ALTER SYSTEM。

schema_name

包含所要列出資料表的結構描述名稱。

若要顯示資料 AWS Glue Data Catalog 表，請提供資 AWS Glue 料庫名稱做為結構描述名稱。

table_name

包含所要列出資料欄的資料表名稱。

filter_pattern

有效的 UTF-8 字元運算式，包含要比對資料表名稱的模式。LIKE 選項會執行區分大小寫的比對，以支援下列模式比對中繼字元：

中繼字元	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。

如果 filter_pattern 未包含任何中繼字元，則模式只代表字串本身，此時 LIKE 的功用如同等於運算子。

row_limit

傳回的最大資料列數。row_limit 可以是 0 到 10,000。

範例

下列範例顯示 Amazon Redshift 資料庫中名為 dev 且在結構描述 public 和資料表 tb 中的資料欄。

```
SHOW COLUMNS FROM TABLE dev.public.tb;

 database_name | schema_name | table_name | column_name | ordinal_position
 | column_default | is_nullable | data_type | character_maximum_length |
 numeric_precision | remarks
---------------+-------------+------------+-------------+------------------
+----------------+-------------+-----------+-------------------------
+-------------------+---------
 dev           | public      | tb         | col         |                1 |
     | YES         | integer   |                         |                32 |
```

下面的例子顯示了在模式和awsdatacatalog表名為 AWS Glue Data Catalog 數據庫中batman的表nation。輸出限制為 2 個資料列。

```
SHOW COLUMNS FROM TABLE awsdatacatalog.batman.nation LIMIT 2;

 database_name  | schema_name | table_name | column_name | ordinal_position
 | column_default | is_nullable | data_type | character_maximum_length |
 numeric_precision | remarks
----------------+-------------+------------+-------------+------------------
+----------------+-------------+-----------+-------------------------
+-------------------+---------
 awsdatacatalog | batman      | nation     | n_nationkey |                1 |
      |             | integer   |                         |                  |
 awsdatacatalog | batman      | nation     | n_name      |                2 |
      |             | character |                         |                  |
```

SHOW EXTERNAL TABLE

顯示外部資料表的定義，包括資料表屬性和資料欄屬性。您可以使用 SHOW EXTERNAL TABLE 陳述式的輸出來重新建立資料表。

如需外部資料表建立的相關資訊，請參閱 CREATE EXTERNAL TABLE。

語法

```
SHOW EXTERNAL TABLE [external_database].external_schema.table_name [ PARTITION ]
```

參數

external_database

相關聯外部資料庫的名稱。此為選用參數。

external_schema

相關聯外部結構描述的名稱。

table_name

要顯示的資料表名稱。

PARTITION

顯示 ALTER TABLE 敘述句，以將分割區新增至資料表定義。

範例

以下範例基於外部資料表定義如下：

```
CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (
     csmallint smallint,
     cint int,
     cbigint bigint,
     cfloat float4,
     cdouble float8,
     cchar char(10),
     cvarchar varchar(255),
     cdecimal_small decimal(18,9),
     cdecimal_big decimal(30,15),
     ctimestamp TIMESTAMP,
     cboolean boolean,
     cstring varchar(16383)
)
PARTITIONED BY (cdate date, ctime TIMESTAMP)
```

```
STORED AS PARQUET
LOCATION 's3://mybucket-test-copy/alldatatypes_parquet_partitioned';
```

以下是 SHOW EXTERNAL TABLE 命令的範例，以及資料表 my_schema.alldatatypes_parquet_test_partitioned 的輸出。

```
SHOW EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned;
```

```
"CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (
    csmallint smallint,
    cint int,
    cbigint bigint,
    cfloat float4,
    cdouble float8,
    cchar char(10),
    cvarchar varchar(255),
    cdecimal_small decimal(18,9),
    cdecimal_big decimal(30,15),
    ctimestamp timestamp,
    cboolean boolean,
    cstring varchar(16383)
)
PARTITIONED BY (cdate date, ctime timestamp)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://mybucket-test-copy/alldatatypes_parquet_partitioned';"
```

以下是 SHOW 外部表命令和輸出為同一個表的一個例子，但在參數中也指定了數據庫。

```
SHOW EXTERNAL TABLE my_database.my_schema.alldatatypes_parquet_test_partitioned;
```

```
"CREATE EXTERNAL TABLE my_database.my_schema.alldatatypes_parquet_test_partitioned (
    csmallint smallint,
    cint int,
    cbigint bigint,
    cfloat float4,
    cdouble float8,
    cchar char(10),
    cvarchar varchar(255),
```

```
    cdecimal_small decimal(18,9),
    cdecimal_big decimal(30,15),
    ctimestamp timestamp,
    cboolean boolean,
    cstring varchar(16383)
)
PARTITIONED BY (cdate date, ctime timestamp)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://mybucket-test-copy/alldatatypes_parquet_partitioned';"
```

以下是 SHOW EXTERNAL TABLE 命令的範例，以及使用 PARTITION 參數時的輸出。輸出包含 ALTER TABLE 陳述式，以將分割區新增至資料表定義。

```
SHOW EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned PARTITION;
```

```
"CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (
    csmallint smallint,
    cint int,
    cbigint bigint,
    cfloat float4,
    cdouble float8,
    cchar char(10),
    cvarchar varchar(255),
    cdecimal_small decimal(18,9),
    cdecimal_big decimal(30,15),
    ctimestamp timestamp,
    cboolean boolean,
    cstring varchar(16383)
)
PARTITIONED BY (cdate date)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://mybucket-test-copy/alldatatypes_parquet_partitioned';
ALTER TABLE my_schema.alldatatypes_parquet_test_partitioned ADD IF NOT
 EXISTS PARTITION (cdate='2021-01-01') LOCATION 's3://mybucket-test-copy/
alldatatypes_parquet_partitioned2/cdate=2021-01-01';
ALTER TABLE my_schema.alldatatypes_parquet_test_partitioned ADD IF NOT
 EXISTS PARTITION (cdate='2021-01-02') LOCATION 's3://mybucket-test-copy/
alldatatypes_parquet_partitioned2/cdate=2021-01-02';"
```

SHOW DATABASES

顯示來自指定帳戶 ID 的資料庫。

語法

```
SHOW DATABASES FROM
DATA CATALOG [ ACCOUNT  '<id1>', '<id2>', ... ]
[ LIKE '<expression>' ]
[ IAM_ROLE default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>' ]
```

參數

ACCOUNT '<id1>', '<id2>', ...

要列出資料庫的 AWS Glue Data Catalog 帳戶。省略此參數表示 Amazon Redshift 應該顯示擁有叢集之帳戶中的資料庫。

LIKE '<expression>'

從資料庫清單中篩選符合您指定運算式的資料庫。此參數支援使用萬用字元 % (百分比) 和 _ (底線) 的模式。

IAM_ROLE default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>'

如果您在執行 SHOW DATABASES 命令時指定與叢集關聯的 IAM 角色，當您在資料庫上執行查詢時，Amazon Redshift 將會使用該角色的登入資料。

指定 `default` 關鍵字表示使用設定為預設值且與叢集相關聯的 IAM 角色。

如果您使用聯合身分連線到 Amazon Redshift 叢集，並從使用 the section called “CREATE DATABASE” 命令建立的外部資料庫存取資料表，請使用 `'SESSION'`。如需使用聯合身分的範例，請參閱使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取，其中會說明如何設定聯合身分。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。IAM 角色最少須具有在所要存取的 Amazon S3 儲存貯體上執行 LIST 操作，以及在儲存貯體包含的 Amazon S3 物件上執行 GET 操作的許可。若要深入瞭解從資料庫建立的資料庫以及使用 IAM_ROLE 的資料庫，請參閱以 AWS Glue Data Catalog 取用者身分使用 Lake 格式化管理的資料庫。

以下顯示單一 ARN 的 IAM_ROLE 參數字串語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

您可以鏈結角色，以便您的叢集可以擔任其他 IAM 角色 (可能屬於其他帳戶)。您最多可以鏈結 10 個角色。如需詳細資訊，請參閱 在 Amazon Redshift Spectrum 中鏈結 IAM 角色。

對於此 IAM 角色，請附加與以下內容相似的 IAM 許可政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessSecret",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetResourcePolicy",
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:ListSecretVersionIds"
            ],
            "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-
rds-secret-VNenFy"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        }
    ]
}
```

如需建立 IAM 角色以搭配聯合查詢使用的步驟，請參閱建立秘密和 IAM 角色來使用聯合查詢。

> **Note**
>
> 不要在鏈結的角色清單中包含空格。

以下顯示鏈結三個角色的語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-
account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

範例

下列範例會顯示帳戶識別碼 123456789012 中的所有 Data Catalog 資料庫。

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012'

  catalog_id  | database_name |                          database_arn
 |     type     |                                          target_database
                            | location | parameters
--------------+---------------+-----------------------------------------------------------
+--------------
+-------------------------------------------------------------------------------------------
+----------+------------
 123456789012 |   database1   | arn:aws:glue:us-east-1:123456789012:database/database1
 | Data Catalog |
                            |          |
 123456789012 |   database2   | arn:aws:glue:us-east-1:123456789012:database/database2
 | Data Catalog | arn:aws:redshift:us-
east-1:123456789012:datashare:035c45ea-61ce-86f0-8b75-19ac6102c3b7/database2 |
 |
```

以下範例會示範如何在使用 IAM 角色的登入資料時，顯示帳戶識別碼 123456789012 中的所有 Data Catalog 資料庫。

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012' IAM_ROLE default;
```

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012' IAM_ROLE <iam-role-arn>;
```

SHOW MODEL

顯示有關機器學習模型的實用資訊，包括其狀態、用於建立模型的參數，以及具有輸入引數類型的預測函數。您可以使用 SHOW MODEL 中的資訊來重新建立模型。如果基底資料表已變更，使用相同的 SQL 陳述式執行 CREATE MODEL 會產生不同的模型。由 SHOW MODEL 傳回的資訊與模型

所有者和具有 EXECUTE 權限的使用者不同。從 Amazon Redshift 訓練模型或模型為 BYOM 模型時，SHOW MODEL 會顯示不同的輸出。

語法

```
SHOW MODEL ( ALL | model_name )
```

參數

ALL

傳回使用者可以使用的所有模型及其結構描述。

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

使用須知

SHOW MODEL 會傳回下列結果：

- 模型名稱。
- 建立模型所在的結構描述。
- 模型的擁有者。
- 模型建立時間。
- 模型的狀態，例如 READY、TRAINING 或 FAILED。
- 失敗模型的原因訊息。
- 如果模型已完成訓練，則會出現驗證錯誤。
- 衍生非 BYOM 方法之模型所需的估計成本。只有模型的擁有者可以檢視此資訊。
- 使用者指定的參數及其值的清單，特別是下列各項：
  - 指定的 TARGET 資料欄。
  - 模型類型 AUTO 或 XGBoost。
  - 問題類型，例如 REGRESSION、BINARY_CLASSIFICATION、MULTICLASS_CLASSIFICATION。此參數是 AUTO 特有的參數。
  - Amazon SageMaker 培訓任務的名稱或創建模型的 Amazon SageMaker 自動駕駛儀任務。您可以使用此工作名稱在 Amazon 上查找有關該模型的更多信息 SageMaker。

- 目標，如 MSE、F1、準確性。此參數是 AUTO 特有的參數。
- 所建立函數的名稱。
- 推論類型 (本機或遠端)。
- 預測函數輸入引數。
- 非使用自有模型 (BYOM) 之模型的預測函數輸入引數類型。
- 預測函數的傳回類型。此參數是 BYOM 特有的參數。
- 具有遠 SageMaker 端推論之 BYOM 模型的 Amazon 端點名稱。
- IAM 角色。只有模型的擁有者可以看到此項目。
- 使用的 S3 儲存貯體。只有模型的擁有者可以看到此項目。
- AWS KMS 鑰匙，如果有提供。只有模型的擁有者可以看到此項目。
- 模型可以執行的時間上限。

- 如果模型類型不是 AUTO，則 Amazon Redshift 也會顯示所提供超參數及其值的清單。

您還可以在其他目錄資料表中檢視由 SHOW MODEL 提供的一些資訊，例如 pg_proc。Amazon Redshift 會傳回 pg_proc 目錄資料表中註冊的預測函數相關資訊。此資訊包括用於預測函數的輸入引數名稱及其類型。Amazon Redshift 會傳回 SHOW MODEL 命令中的相同資訊。

```
SELECT * FROM pg_proc WHERE proname ILIKE '%<function_name>%';
```

範例

下例範例顯示「顯示模型」的輸出。

```
SHOW MODEL ALL;

Schema Name |  Model Name
------------+---------------
 public     | customer_churn
```

customer_churn 的擁有者可以看到以下輸出。僅具有 EXECUTE 權限的使用者無法看到 IAM 角色、Amazon S3 儲存貯體和模式的預估成本。

```
SHOW MODEL customer_churn;

       Key                  |           Value
----------------------------+---------------------------------
```

```
Model Name                | customer_churn
Schema Name               | public
Owner                     | 'owner'
Creation Time             | Sat, 15.01.2000 14:45:20
Model State               | READY
validation:F1             | 0.855
Estimated Cost            | 5.7
                          |
TRAINING DATA:            |
Table                     | customer_data
Target Column             | CHURN
                          |
PARAMETERS:               |
Model Type                | auto
Problem Type              | binary_classification
Objective                 | f1
Function Name             | predict_churn
Function Parameters       | age zip average_daily_spend average_daily_cases
Function Parameter Types  | int int float float
IAM Role                  | 'iam_role'
KMS Key                   | 'kms_key'
Max Runtime               | 36000
```

SHOW DATASHARES

顯示叢集中來自相同帳戶或跨帳戶的輸入和輸出共用。如果您未指定資料共用名稱，Amazon Redshift 會顯示叢集中所有資料庫中的所有資料共用。具有 ALTER 和 SHARE 權限的使用者可以查看他們有權限的共用。

語法

```
SHOW DATASHARES [ LIKE 'namepattern' ]
```

參數

LIKE

選用子句，將指定的名稱模式與資料共用的描述進行比較。使用此子句時，Amazon Redshift 只會顯示名稱符合指定名稱模式的資料共用。

namepattern

請求的資料共用名稱或名稱的一部分會使用萬用字元進行比對。

範例

下列範例會顯示叢集中的輸入和輸出共用。

```
SHOW DATASHARES;
SHOW DATASHARES LIKE 'sales%';

share_name   | share_owner | source_database | consumer_database | share_type |
 createdate          | is_publicaccessible | share_acl | producer_account |
 producer_namespace
-------------+-------------+-----------------+-------------------+------------
+---------------------+---------------------+-----------+------------------
+--------------------------------------
'salesshare' | 100         | dev             |                   | outbound
 | 2020-12-09 01:22:54.| False               |           |   123456789012   |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

SHOW PROCEDURE

顯示特定預存程序的定義，包括其簽章。您可以使用 SHOW PROCEDURE 的輸出來重新建立預存程序。

語法

```
SHOW PROCEDURE sp_name [( [ [ argname ] [ argmode ] argtype [, ...] ] )]
```

參數

sp_name

要顯示的程序的名稱。

[argname] [argmode] argtype

用來識別預存程序的輸入引數類型。您可以選擇性包含完整引數資料類型，包括 OUT 引數。如果預存程序的名稱是唯一的 (亦即不過載)，則此為選用部分。

範例

下列範例顯示程序 `test_spl2` 的定義。

```
show procedure test_sp2(int, varchar);
```

```
                                        Stored Procedure Definition
-------------------------------------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE public.test_sp2(f1 integer, INOUT f2 character varying, OUT
 character varying)
LANGUAGE plpgsql
AS $_$
DECLARE
out_var alias for $3;
loop_var int;
BEGIN
IF f1 is null OR f2 is null THEN
RAISE EXCEPTION 'input cannot be null';
END IF;
CREATE TEMP TABLE etl(a int, b varchar);
FOR loop_var IN 1..f1 LOOP
insert into etl values (loop_var, f2);
f2 := f2 || '+' || f2;
END LOOP;
SELECT INTO out_var count(*) from etl;
END;
$_$

(1 row)
```

SHOW SCHEMAS

顯示資料庫中的結構描述清單，以及一些結構描述屬性。

每個輸出資料列都包含資料庫名稱、結構描述名稱、結構描述擁有者、結構描述類型、結構描述 ACL、來源資料庫和結構描述選項。如需這些屬性的相關資訊，請參閱 SVV_ALL_SCHEMAS。

如果 SHOW SCHEMA 命令可能產生超過 10,000 個結構描述，則會傳回錯誤。

語法

```
SHOW SCHEMAS FROM DATABASE database_name [LIKE 'filter_pattern'] [LIMIT row_limit ]
```

參數

database_name

包含所要列出資料表的資料庫名稱。

若要在中顯示表格 AWS Glue Data Catalog，請指定 (awsdatacatalog) 作為資料庫名稱，並確定系統組態data_catalog_auto_mount設定為true。如需詳細資訊，請參閱 ALTER SYSTEM。

filter_pattern

有效的 UTF-8 字元運算式，包含要比對結構描述名稱的模式。LIKE 選項會執行區分大小寫的比對，以支援下列模式比對中繼字元：

中繼字元	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。

如果 filter_pattern 未包含任何中繼字元，則模式只代表字串本身，此時 LIKE 的功用如同等於運算子。

row_limit

傳回的最大資料列數。row_limit 可以是 0 到 10,000。

範例

下面的範例會顯示名為 dev 的 Amazon Redshift 資料庫的結構描述。

```
SHOW SCHEMAS FROM DATABASE dev;

 database_name |     schema_name      | schema_owner | schema_type |         schema_acl
          | source_database | schema_option
---------------+----------------------+--------------+-------------
+----------------------------+-----------------+--------------
 dev           | pg_automv            |            1 | local       |
          |                 |
 dev           | pg_catalog           |            1 | local       | jpuser=UC/
jpuser~=U/jpuser  |                 |
 dev           | public               |            1 | local       | jpuser=UC/
jpuser~=UC/jpuser |                 |
 dev           | information_schema   |            1 | local       | jpuser=UC/
jpuser~=U/jpuser  |                 |
```

```
 dev           | schemad79cd6d93bf043 |           1 | local       |
         |                 |
```

下面的例子顯示了名為 AWS Glue Data Catalog 數據庫中的模式awsdatacatalog。輸出資料列數目上限為 5。

```
SHOW SCHEMAS FROM DATABASE awsdatacatalog LIMIT 5;

 database_name  |     schema_name      | schema_owner | schema_type | schema_acl |
 source_database | schema_option
----------------+----------------------+--------------+-------------+------------
+-----------------+---------------
 awsdatacatalog | 000_too_many_glue_db |              | EXTERNAL    |            |
            |
 awsdatacatalog | 123_default          |              | EXTERNAL    |            |
            |
 awsdatacatalog | adhoc                |              | EXTERNAL    |            |
            |
 awsdatacatalog | all_shapes_10mb      |              | EXTERNAL    |            |
            |
 awsdatacatalog | all_shapes_1g        |              | EXTERNAL    |            |
            |
```

SHOW TABLE

顯示資料表的定義，包括資料表屬性、資料表條件限制、資料欄屬性和資料欄條件限制。您可以使用 SHOW TABLE 陳述式的輸出來重新建立資料表。

如需建立資料表的相關資訊，請參閱 CREATE TABLE。

語法

```
SHOW TABLE [schema_name.]table_name
```

參數

schema_name

(選擇性) 相關結構描述的名稱。

table_name

要顯示的資料表名稱。

範例

以下是資料表 sales 的 SHOW TABLE 輸出範例。

```
show table sales;
```

```
CREATE TABLE public.sales (
salesid integer NOT NULL ENCODE az64,
listid integer NOT NULL ENCODE az64 distkey,
sellerid integer NOT NULL ENCODE az64,
buyerid integer NOT NULL ENCODE az64,
eventid integer NOT NULL ENCODE az64,
dateid smallint NOT NULL,
qtysold smallint NOT NULL ENCODE az64,
pricepaid numeric(8,2) ENCODE az64,
commission numeric(8,2) ENCODE az64,
saletime timestamp without time zone ENCODE az64
)
DISTSTYLE KEY SORTKEY ( dateid );
```

以下是結構描述 public 中資料表 category 的 SHOW TABLE 輸出範例。

```
show table public.category;
```

```
CREATE TABLE public.category (
catid smallint NOT NULL distkey,
catgroup character varying(10) ENCODE lzo,
catname character varying(10) ENCODE lzo,
catdesc character varying(50) ENCODE lzo
) DISTSTYLE KEY SORTKEY ( catid );
```

下列範例會建立具有主索引鍵的資料表 foo。

```
create table foo(a int PRIMARY KEY, b int);
```

SHOW TABLE 結果會顯示含有 foo 資料表所有屬性的建立陳述式。

```
show table foo;
```

```
CREATE TABLE public.foo ( a integer NOT NULL ENCODE az64, b integer ENCODE az64,
 PRIMARY KEY (a) ) DISTSTYLE AUTO;
```

SHOW TABLES

顯示結構描述中的資料表清單，以及一些資料表屬性。

每個輸出資料列都包含資料庫名稱、結構描述名稱、資料表名稱、資料表類型、資料表 ACL 和備註。如需這些屬性的相關資訊，請參閱 SVV_ALL_TABLES。

如果 SHOW TABLES 命令所產生的資料表超過 10,000 個，則會傳回錯誤。

語法

```
SHOW TABLES FROM SCHEMA database_name.schema_name [LIKE 'filter_pattern']
 [LIMIT row_limit ]
```

參數

database_name

包含所要列出資料表的資料庫名稱。

若要在中顯示表格 AWS Glue Data Catalog，請指定 (awsdatacatalog) 作為資料庫名稱，並確定系統組態data_catalog_auto_mount設定為true。如需詳細資訊，請參閱 ALTER SYSTEM。

schema_name

包含所要列出資料表的結構描述名稱。

若要顯示資料 AWS Glue Data Catalog 表，請提供資 AWS Glue 料庫名稱做為結構描述名稱。

filter_pattern

有效的 UTF-8 字元運算式，包含要比對資料表名稱的模式。LIKE 選項會執行區分大小寫的比對，以支援下列模式比對中繼字元：

中繼字元	描述
%	比對任何 0 的序列或更多字元。

中繼字元	描述
_	比對任一個單一字元。

如果 filter_pattern 未包含任何中繼字元，則模式只代表字串本身，此時 LIKE 的功用如同等於運算子。

row_limit

傳回的最大資料列數。row_limit 可以是 0 到 10,000。

範例

下列範例顯示 Amazon Redshift 資料庫中名為 dev 且在結構描述 public 中的資料表。

```
SHOW TABLES FROM SCHEMA dev.public;

 database_name | schema_name | table_name | table_type | table_acl | remarks
---------------+-------------+------------+------------+-----------+---------
 dev           | public      | tb         | TABLE      |           |
 dev           | public      | tb2        | TABLE      |           |
 dev           | public      | tb3        | TABLE      |           |
```

下面的例子顯示了在模式名awsdatacatalog為 AWS Glue Data Catalog 數據庫中的表batman。

```
SHOW TABLES FROM SCHEMA awsdatacatalog.batman;

 database_name  | schema_name |    table_name    | table_type | table_acl | remarks
----------------+-------------+------------------+------------+-----------+---------
 awsdatacatalog | batman      | nation           | EXTERNAL   |           |
 awsdatacatalog | batman      | part             | EXTERNAL   |           |
 awsdatacatalog | batman      | partsupp         | EXTERNAL   |           |
 awsdatacatalog | batman      | region           | EXTERNAL   |           |
 awsdatacatalog | batman      | supplier         | EXTERNAL   |           |
 awsdatacatalog | batman      | automount_nation | EXTERNAL   |           |
```

SHOW VIEW

顯示檢視的定義，包括具體化視觀表和近期繫結檢視。您可以使用 SHOW VIEW 陳述式的輸出來重新建立檢視。

語法

```
SHOW VIEW [schema_name.]view_name
```

參數

schema_name

(選擇性) 相關結構描述的名稱。

view_name

要顯示的檢視名稱。

範例

以下為 LA_Venues_v 檢視的檢視定義：

```
create view LA_Venues_v as select * from venue where venuecity='Los Angeles';
```

以下範例顯示 SHOW VIEW 命令和先前所定義檢視的輸出。

```
show view LA_Venues_v;
```

```
SELECT venue.venueid,
venue.venuename,
venue.venuecity,
venue.venuestate,
venue.venueseats
FROM venue WHERE ((venue.venuecity)::text = 'Los Angeles'::text);
```

以下為結構描述 public 中檢視 public.Sports_v 的檢視定義。

```
create view public.Sports_v as select * from category where catgroup='Sports';
```

以下範例顯示 SHOW VIEW 命令和先前所定義檢視的輸出。

```
show view public.Sports_v;
```

```
SELECT category.catid,
category.catgroup,
category.catname,
category.catdesc
FROM category WHERE ((category.catgroup)::text = 'Sports'::text);
```

START TRANSACTION

BEGIN 函數的同義詞。

請參閱BEGIN。

TRUNCATE

刪除資料表中的所有資料列，但不執行資料表掃描：此操作是速度比非限定 DELETE 操作更快的替代方案。若要執行 TRUNCATE 命令，您必須具有 TRUNCATE TABLE 權限、是資料表的擁有者或超級使用者。若要授予截斷資料表的權限，請使用 GRANT 命令。

TRUNCATE 比 DELETE 更有效率，而且不需要 VACUUM 和 ANALYZE。不過請注意，TRUNCATE 會遞交其執行所在的交易。

語法

```
TRUNCATE [ TABLE ] table_name
```

此命令也適用於具體化視觀表。

```
TRUNCATE materialized_view_name
```

參數

TABLE

選用的關鍵字。

table_name

暫時性或持久性資料表。只有資料表的擁有者或超級使用者可將它截斷。

您可以截斷任何資料表，包括外部索引鍵限制條件中參考的資料表。

您不需要在截斷資料表之後將它清空。

materialized_view_name

具體化視觀表。

您可以截斷用於 串流擷取 的具體化視觀表。

使用須知

TRUNCATE 命令會遞交其執行所在的交易；因此您無法轉返 TRUNCATE 操作，且 TRUNCATE 命令可能會在遞交本身的同時，遞交其他操作。

範例

使用 TRUNCATE 命令可刪除 CATEGORY 資料表中的所有資料列：

```
truncate category;
```

嘗試轉返 TRUNCATE 操作：

```
begin;

truncate date;

rollback;

select count(*) from date;
count
-------
0
(1 row)
```

DATE 資料表會在 ROLLBACK 命令之後保持空白，因為 TRUNCATE 命令會自動遞交。

下列範例會使用 TRUNCATE 命令，刪除具體化視觀表中的所有資料列。

```
truncate my_materialized_view;
```

它會刪除具體化視觀表中的所有記錄，並保持具體化視觀表及其結構描述不變。在查詢中，具體化視觀表名稱是一個範例。

UNLOAD

使用 Amazon S3 伺服器端加密 (SSE-S3)，將查詢的結果卸載至 Amazon S3 上的一或多個文字檔案、JSON 檔案或 Apache Parquet 檔案。您也可以指定伺服器端加密搭配 AWS Key Management Service 金鑰 (SSE-KMS)，或用戶端加密搭配客戶受管金鑰。

依預設，卸載檔案的格式為以豎線分隔 (|) 的文字。

您可以設定 MAXFILESIZE 參數來管理 Amazon S3 上檔案的大小，以及藉由副檔名管理檔案數目。確定 S3 IP 範圍已新增至您的允許清單。若要進一步了解所需的 S3 IP 範圍，請參閱網路隔離。

您可以將 Amazon Redshift 查詢的結果以 Apache Parquet 卸載至您的 Amazon S3 資料湖，這是一種用於分析的高效率開放單欄式儲存格式。相較於文字格式，Parquet 格式的卸載速度提升達 2 倍，並且在 Amazon S3 中使用的儲存空間減少達 6 倍。這可以讓您將已在 Amazon S3 中執行的資料轉換和資料充實，以開放的格式儲存到您的 Amazon S3 資料湖。然後，您可以使用 Redshift 頻譜和其他 AWS 服務（例如 Amazon 雅典娜，Amazon EMR 和亞馬遜）來分析數據。 SageMaker

若要取得有關使用 UNLOAD 命令的更多資訊和範例案例，請參閱 卸載資料。

所需的權限和許可

為了使 UNLOAD 命令成功執行，至少需要資料庫中資料的 SELECT 權限，以及寫入 Amazon S3 位置的許可。所需的許可類似於 COPY 命令。如需有關 COPY 命令的詳細資訊，請參閱 存取其他 AWS 資源的許可。

語法

```
UNLOAD ('select-statement')
TO 's3://object-path/name-prefix'
authorization
[ option, ...]

where authorization is
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id-1>:role/<role-name>[,arn:aws:iam::<AWS #
#-id-2>:role/<role-name>][,...]' }

where option is
| [ FORMAT [ AS ] ] CSV | PARQUET | JSON
| PARTITION BY ( column_name [, ... ] ) [ INCLUDE ]
| MANIFEST [ VERBOSE ]
| HEADER
```

```
| DELIMITER [ AS ] 'delimiter-char'
| FIXEDWIDTH [ AS ] 'fixedwidth-spec'
| ENCRYPTED [ AUTO ]
| BZIP2
| GZIP
| ZSTD
| ADDQUOTES
| NULL [ AS ] 'null-string'
| ESCAPE
| ALLOWOVERWRITE
| CLEANPATH
| PARALLEL [ { ON | TRUE } | { OFF | FALSE } ]
| MAXFILESIZE [AS] max-size [ MB | GB ]
| ROWGROUPSIZE [AS] size [ MB | GB ]
| REGION [AS] 'aws-region' }
| EXTENSION 'extension-name'
```

參數

('select-statement')

SELECT 查詢。查詢的結果會卸載。大多數情況下，藉由在查詢中指定 ORDER BY 子句依排序順序卸載資料會有其實用性。此方式可節省重新載入資料時，排序資料所需的時間。

查詢必須用單引號括住，如下所示：

```
('select * from venue order by venueid')
```

> **Note**
>
> 如果您的查詢包含引號 (例如為了將常值括住)，請將常值放在兩組單引號內，您也必須將查詢放在單引號之間：
>
> ```
> ('select * from venue where venuestate=''NV''')
> ```

TO 's3://object-path/name-prefix'

Amazon S3 上 Amazon Redshift 將寫入輸出檔案物件之位置的完整路徑 (包括儲存貯體名稱在內)，若指定了 MANIFEST，則包括資訊清單檔案。物件名稱前面會加上 name-prefix。如果您使

用 PARTITION BY，會自動視需要將一個斜線 (/) 新增到 name-prefix 值的結尾。為了提升安全性，UNLOAD 會使用 HTTPS 連線連接至 Amazon S3。根據預設，UNLOAD 會在每個分割中寫入一個或多個檔案。UNLOAD 會在指定的名稱字首附加分割號碼和部分編號，如下所示：

<object-path>/*<name-prefix>*<*slice-number*>_part_<*part-number*>.

若指定了 MANIFEST，則會寫入資訊清單檔案，如下所示：

<object_path>/*<name_prefix>*manifest.

如果指定 PARALLEL 為 OFF，資料檔案會寫入如下：

<object_path>/*<name_prefix>*<*part-number*>.

UNLOAD 會使用 Amazon S3 伺服器端加密 (SSE) 自動建立加密檔案，若使用了 MANIFEST，則也會包括資訊清單檔案。COPY 命令會在載入操作期間自動讀取伺服器端加密檔案。您可以使用 Amazon S3 主控台或 API，透明地從儲存貯體下載伺服器端加密檔案。如需詳細資訊，請參閱使用伺服器端加密保護資料。

若要使用 Amazon S3 用戶端加密，請指定 ENCRYPTED 選項。

> **Important**
>
> 當 Amazon S3 儲存貯體與 Amazon Redshift 資料庫不在相同 AWS 區域 內時，就需要 REGION。

authorization

UNLOAD 命令需要授權才能將資料寫入 Amazon S3。UNLOAD 命令會使用與 COPY 命令相同的參數進行授權。如需詳細資訊，請參閱 COPY 命令語法參考中的 授權參數。

IAM_ROLE { default | 'arn:aws:iam::*<AWS ##-id-1>*:role/*<role-name>*'

使用預設關鍵字，讓 Amazon Redshift 使用設定為預設並在執行 UNLOAD 命令時與叢集關聯的 IAM 角色。

對叢集進行身分驗證和授權時所使用的 IAM 角色使用 Amazon Resource Name (ARN)。如果指定 IAM_ROLE，則不能使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY、SESSION_TOKEN 或 CREDENTIALS。IAM_ROLE 可以鏈結。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的鏈結 IAM 角色。

[FORMAT [AS]] CSV | PARQUET | JSON

關鍵字，用來指定取代預設格式的卸載格式。

使用 CSV 時，會使用逗號 (,) 字元做為預設分隔符號卸載至 CSV 格式的文字檔案。如果欄位包含分隔符號、雙引號、新行字元或歸位字元，已卸載檔案中的欄位便會括在雙引號中。資料欄位中的雙引號會使用額外的雙引號進行逸出。卸載零個資料列時，Amazon Redshift 可能會寫入空的 Amazon S3 物件。

使用 PARQUET 時，會卸載至 Apache Parquet 1.0 版格式的檔案。根據預設，每個資料列群組都會使用 SNAPPY 壓縮進行壓縮。如需 Apache Parquet 格式的相關資訊，請參閱 Parquet。

使用 JSON 時會卸載到 JSON 文件，每行都會包含一個 JSON 物件，代表查詢結果中的完整記錄。當查詢結果包含 SUPER 資料欄時，Amazon Redshift 支援編寫巢狀 JSON。若要建立有效的 JSON 物件，查詢中每個資料欄的名稱必須是唯一的。在 JSON 文件中，布林值會卸載為 `t` 或 `f`，NULL 值會被卸載為 `null`。卸載零個資料列時，Amazon Redshift 不會寫入 Amazon S3 物件。

FORMAT 和 AS 關鍵字是選用的。您無法使用 CSV 與 FIXEDWIDTH 或 ADDQUOTES 搭配。您無法搭配 DELIMITER、FIXEDWIDTH、ADDQUOTES、ESCAPE、NULL AS、HEADER、GZIP、BZIP2 或 ZSTD 使用 PARQUET。只有使用金 AWS Key Management Service 鑰 (SSE-KMS) 的伺服器端加密才支援加密的實木複合地板。您無法搭配 DELIMITER、HEADER、FIXEDWIDTH、ADDQUOTES、ESCAPE 或 NULL AS 使用 JSON。

PARTITION BY (column_name [, ...]) [INCLUDE]

指定卸載操作的分割區索引鍵。UNLOAD 會依序根據分割區索引鍵的值和 Apache Hive 慣例，自動將輸出檔案分到分割資料夾。例如，屬於 2019 年分割區和九月分割區的 Parquet 檔案便會具有以下字首：`s3://my_bucket_name/my_prefix/year=2019/month=September/000.parquet`。

column_name 的值必須是正在卸載查詢結果中的資料行。

如果您指定 PARTITION BY 搭配 INCLUDE 選項，則不會從上傳的檔案中移除分割區資料欄。

Amazon Redshift 不支援 PARTITION BY 子句中的字串文字。

MANIFEST [VERBOSE]

會建立資訊清單檔案，當中明確列出 UNLOAD 程序建立的資料檔案的詳細資訊。清單檔案是 JSON 格式的文字檔，其中列出寫入 Amazon S3 的每個檔案的 URL。

如果指定 MANIFEST 時搭配 VERBOSE 選項，則資訊清單會包含下列詳細資訊：

- 資料欄名稱和資料類型，以及針對 CHAR、VARCHAR 或 NUMERIC 資料類型，每個資料欄的維度。針對 CHAR 和 VARCHAR 資料類型，維度為長度。針對 DECIMAL 或 NUMERIC 資料類型，維度為精確度和規模。
- 卸載至每個檔案的資料列計數。如果指定了 HEADER 選項，則資料列計數會包括標題行。
- 卸載的所有檔案的檔案大小總計以及卸載至所有檔案的資料列計數總計。如果指定了 HEADER 選項，則資料列計數會包括標題行。
- 作者。作者一律是 "Amazon Redshift"。

您只能在 MANIFEST 之後指定 VERBOSE。

資訊清單檔案會寫入與卸載檔案相同的 Amazon S3 路徑字首，其格式為 `<object_path_prefix>manifest`。例如，若 UNLOAD 指定 Amazon S3 路徑字首 `'s3://mybucket/venue_'`，則資訊清單檔案的位置會是 `'s3://mybucket/venue_manifest'`。

HEADER

在每個輸出檔案上方新增包含資料欄名稱的標題行。文字轉換選項，例如 CSV、DELIMITER、ADDQUOTES 和 ESCAPE，也適用標題行。您無法使用 HEADER 與 FIXEDWIDTH 搭配。

DELIMITER AS 'delimiter_character'

指定單一 ASCII 字元，用以分隔輸出檔案中的欄位，例如縱線字元 (|)、逗號 (,) 或 Tab 字元 (\t)。文字檔的預設分隔符號為縱線字元。CSV 檔的預設分隔符號為逗號字元。AS 關鍵字為選用。您無法使用 DELIMITER 與 FIXEDWIDTH 搭配。若資料包含分隔符號字元，您需要指定 ESCAPE 選項來逸出分隔符號，或使用 ADDQUOTES 將資料括在雙引號中。或者，指定資料中未包含的分隔符號。

FIXEDWIDTH 'fixedwidth_spec'

將資料卸載至每個欄寬是固定長度 (而不是以分隔字元隔開) 的檔案。fixedwidth_spec 是字串，指定資料欄數和欄寬。AS 關鍵字為選用。由於 FIXEDWIDTH 不會截斷資料，因此 UNLOAD 陳述式中每個資料欄的規格都必須至少為該資料欄最長項目的長度。fixedwidth_spec 的格式如下所示：

```
'colID1:colWidth1,colID2:colWidth2, ...'
```

您無法使用 FIXEDWIDTH 與 DELIMITER 或 HEADER 搭配。

ENCRYPTED [AUTO]

指定 Amazon S3 上的輸出檔案將使用 Amazon S3 伺服器端加密或用戶端加密進行加密。若指定了 MANIFEST，則也會加密資訊清單檔案。如需詳細資訊，請參閱 卸載加密的資料檔案。如果您未指

定加密參數，UNLOAD 會使用 Amazon S3 伺服器端加密搭配 AWS受管加密金鑰自動建立加密檔案 (SSE-S3)。

對於加密，您可能想要使用具有 AWS KMS 金鑰 (SSE-KMS) 的伺服器端加密將其卸載到 Amazon S3。如果是這種情況，請使用 KMS_KEY_ID 參數來提供金鑰 ID。您無法使用 CREDENTIALS 參數搭配 KMS_KEY_ID 參數。如果您使用 KMS_KEY_ID 針對資料執行 UNLOAD 命令，您接著便可以針對相同的資料執行 COPY 操作，而無須指定金鑰。

如要搭配客戶提供的對稱金鑰使用用戶端加密卸載至 Amazon S3，請透過以下兩種方式中的其中一種來提供金鑰。如要提供金鑰，請使用 MASTER_SYMMETRIC_KEY 參數，或是 CREDENTIALS 登入資料字串的 `master_symmetric_key` 部分。如果您使用根對稱金鑰卸載資料，請務必在針對加密資料執行 COPY 操作時提供相同的金鑰。

UNLOAD 不支援 Amazon S3 伺服器端加密搭配客戶提供的金鑰 (SSE-C)。

如果使用加密自動，UNLOAD 命令會擷取目標 Amazon S3 儲存貯體屬性上的預設 AWS KMS 加密金鑰，並使用金鑰加密寫入 Amazon S3 的檔案。 AWS KMS 如果儲存貯體沒有預設 AWS KMS 加密金鑰，UNLOAD 會使用 Amazon Redshift 伺服器端加密搭配 AWS受管加密金鑰自動建立加密檔案 (SSE-S3)。您無法搭配 KMS_KEY_ID、MASTER_SYMMETRIC_KEY 或包含 master_symmetric_key 的 CREDENTIALS 使用此選項。

KMS_KEY_ID 'key-id'

指定要用於在 Amazon S3 上加密資料檔案的 AWS Key Management Service (AWS KMS) 金鑰的金鑰 ID。如需詳細資訊，請參閱什麼是 AWS Key Management Service？ 如果您指定 KMS_KEY_ID，則必須一併指定 ENCRYPTED 參數。如果您指定 KMS_KEY_ID，則不能使用 CREDENTIALS 參數進行驗證。請改用 IAM_ROLE 或 ACCESS_KEY_ID and SECRET_ACCESS_KEY。

MASTER_SYMMETRIC_KEY 'root_key'

指定要用來加密 Amazon S3 上資料檔案的根對稱金鑰。如果您指定 MASTER_SYMMETRIC_KEY，則必須一併指定 ENCRYPTED 參數。您無法使用 MASTER_SYMMETRIC_KEY 與 CREDENTIALS 參數搭配。如需詳細資訊，請參閱 從 Amazon S3 載入加密的資料檔案。

BZIP2

將資料卸載至每個分割中的一個或多個 bzip2 壓縮檔。產生的每個檔案都會附加 `.bz2` 副檔名。

GZIP

將資料卸載至每個分割中的一個或多個 gzip 壓縮檔。產生的每個檔案都會附加 `.gz` 副檔名。

ZSTD

將資料卸載至每個分割中的一個或多個 Zstandard 壓縮檔。產生的每個檔案都會附加 `.zst` 副檔名。

ADDQUOTES

在每個卸載的資料欄位前後加上引號，如此 Amazon Redshift 就能卸載包含分隔符號本身的資料值。例如，如果分隔符號是逗號，您可以成功卸載並重新載入下列資料：

```
"1","Hello, World"
```

如果沒有新增雙引號，`Hello, World` 字串將會剖析為兩個不同欄位。

某些輸出格式不支援 ADDQUOTES。

若您使用 ADDQUOTES，則必須在重新載入資料時，於 COPY 中指定 REMOVEQUOTES。

NULL AS 'null-string'

指定一個字串，代表卸載檔案中的 null 值。若使用此選項，所有輸出檔案都會包含指定的字串，用以取代所選取資料中找到的任何 null 值。若未指定此選項，則 null 值會卸載為：

- 分隔符號輸出的零長度字串
- 固定寬度輸出的空白字串

若針對固定寬度卸載指定 null 字串，而輸出資料欄的寬度小於 null 字串的寬度，則會發生下列行為：

- 非字元資料欄的輸出為空欄位
- 針對字元資料欄回報錯誤

與其他使用者定義字串代表 null 值的資料類型不同，Amazon Redshift 會使用 JSON 格式匯出 SUPER 資料欄，並在 JSON 格式決定時將其表示為 null 值。因此，SUPER 資料欄會忽略 UNLOAD 命令中使用的 NULL [AS] 選項。

ESCAPE

分隔符號卸載檔案的 CHAR 和 VARCHAR 資料欄中會放入逸出字元 (\)，位於出現下列字元的每個位置前面：

- 換行：`\n`
- 歸位字元: `\r`

- 針對卸載資料指定的分隔符號字元。
- 逸出字元：\
- 引號字元：" 或 ' (如果 UNLOAD 命令中同時指定了 ESCAPE 與 ADDQUOTES)。

> **Important**
>
> 如果您使用 COPY 搭配 ESCAPE 選項載入資料，則在 UNLOAD 命令中也必須指定 ESCAPE 選項，以產生具相互關係的輸出檔案。同樣地，如果您使用 ESCAPE 選項執行 UNLOAD，則在 COPY 相同的資料時需要使用 ESCAPE。

ALLOWOVERWRITE

根據預設，若找到可能會覆寫的檔案，則 UNLOAD 會失敗。若指定了 ALLOWOVERWRITE，UNLOAD 會覆寫現有檔案，包括資訊清單檔案。

CLEANPATH

在將檔案卸載到指定位置之前，CLEANPATH 選項會移除 TO 子句中指定之 Amazon S3 路徑中的現有檔案。

如果您包含 PARTITION BY 子句，則只會從分割區資料夾中移除現有檔案，以接收 UNLOAD 作業所產生的新檔案。

您必須具有 Amazon S3 儲存貯體的 `s3:DeleteObject` 許可。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 Amazon S3 的政策和許可。您使用 CLEANPATH 選項移除的檔案會永久刪除，且無法復原。

如果您指定 ALLOWOVERWRITE 選項，則無法指定 CLEANPATH 選項。

PARALLEL

依預設，UNLOAD 會根據叢集中的分割數，將資料平行寫入多個檔案。預設選項為 ON 或 TRUE。若 PARALLEL 為 OFF 或 FALSE，則 UNLOAD 會依序寫入一個或多個資料檔案，並絕對會根據 ORDER BY 子句 (如有使用的話) 排序。資料檔案大小上限為 6.2 GB。因此，假如您卸載 13.4 GB 的資料，UNLOAD 會建立下列三個檔案。

```
s3://mybucket/key000    6.2 GB
s3://mybucket/key001    6.2 GB
s3://mybucket/key002    1.0 GB
```

> **Note**
>
> UNLOAD 命令是設計為使用平行處理。我們建議在多數的情況將 PARALLEL 保持為啟用，特別是在使用 COPY 命令將檔案是用來載入資料表時。

MAXFILESIZE [AS] max-size [MB | GB]

指定 UNLOAD 在 Amazon S3 中建立的檔案大小上限。指定介於 5 MB 和 6.2 GB 之間的小數值。AS 關鍵字為選用。預設單位為 MB。如果未指定 MAXFILESIZE，則預設的檔案大小上限為 6.2 GB。資訊清單檔案的大小 (如有使用的話) 不受 MAXFILESIZE 的影響。

ROWGROUPSIZE [AS] size [MB | GB]

指定資料列群組的大小。選擇較大的大小可減少資料列群組的數目，進而減少網路通訊量。請指定介於 32 MB 到 128 MB 之間的整數值。AS 關鍵字為選用。預設單位為 MB。

若沒有指定 ROWGROUPSIZE，則預設大小為 32 MB。若要使用此參數，儲存格式必須是鑲木地板，且節點類型必須是 ra3.4xlarge、ra3.16xlarge 或 dc2.8xlarge。

REGION [AS] 'aws-region'

指定目標 Amazon S3 儲存貯體所 AWS 區域 在的位置。要卸載到與亞馬遜紅移數 AWS 區域 據庫不同的 Amazon S3 存儲桶，則需要區域。

aws_ 區域的值必須與中 Amazon Redshift AWS 區域和端點資料表中列出的區域相符。AWS 一般參考

根據預設，UNLOAD 會假設目標 Amazon S3 儲存貯體與亞馬 Amazon Redshift 資料庫位於 AWS 區域 相同的位置。

EXTENSION 'extension-name'

指定要附加到卸載檔案名稱的副檔名。Amazon Redshift 不會執行任何驗證，因此您必須確認指定的副檔名是否正確。如果您使用的是 GZIP 之類的壓縮方法，您仍然必須在副檔名參數中指定 `.gz`。如果您不提供任何副檔名，Amazon Redshift 不會在文件名中添加任何內容。如果您指定壓縮方法而不提供副檔名，Amazon Redshift 只會將壓縮方法的副檔名新增至檔案名稱。

使用須知

針對所有分隔文字的 UNLOAD 操作使用 ESCAPE

當您使用分隔符號執行 UNLOAD，則您的資料中可能包括分隔符號或 ESCAPE 選項說明中列出的任何字元。在這種情況下，則您必須在 UNLOAD 陳述式中使用 ESCAPE 選項。若您未使用 ESCAPE 選項搭配 UNLOAD，則後續使用卸載資料的 COPY 操作可能會失敗。

> **Important**
>
> 強烈建議您一律使用 ESCAPE 搭配 UNLOAD 和 COPY 陳述式。如果您確定資料未包含任何分隔符號或可能需要逸出的其他字元，則為例外。

浮點數精確度喪失

您可能遇到後續卸載並重新載入的浮點資料精確度喪失的情況。

Limit 子句

SELECT 查詢無法在外部 SELECT 中使用 LIMIT 子句。例如，以下 UNLOAD 陳述式會失敗。

```
unload ('select * from venue limit 10')
to 's3://mybucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/
MyRedshiftRole';
```

請改為使用巢狀 LIMIT 子句，如下列範例所示。

```
unload ('select * from venue where venueid in
(select venueid from venue order by venueid desc limit 10)')
to 's3://mybucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/
MyRedshiftRole';
```

您也可以使用 SELECT...INTO 或使用 LIMIT 子句的 CREATE TABLE AS 填入資料表，然後從該資料表卸載。

卸載 GEOMETRY 資料類型的欄位

您只能將 GEOMETRY 資料行卸載至文字或 CSV 格式。您無法卸載包含 FIXEDWIDTH 選項的 GEOMETRY 資料。資料會以擴充已知二進位 (EWKB) 格式的十六進位形式卸載。如果 EWKB 資料的大小超過 4 MB，則會發生警告，因為資料稍後將無法載入資料表。

卸載 HLLSKETCH 資料類型

您只能將 HLLSKETCH 資料欄卸載至文字或 CSV 格式。您無法卸載包含 FIXEDWIDTH 選項的 HLLSKETCH 資料。如果是密集 HyperLogLog 草圖，則會以 Base64 格式卸載資料，若是稀疏 HyperLogLog 草圖則以 JSON 格式卸載。如需詳細資訊，請參閱 HyperLogLog 函數。

下列範例會將包含 HLLSKETCH 資料欄的資料表匯出至檔案。

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
 (5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

UNLOAD ('select * from hll_table') TO 's3://mybucket/unload/'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole' NULL AS 'null' ALLOWOVERWRITE
 CSV;
```

卸載 VARBYTE 資料類型的資料欄

您只能將 VARBYTE 資料欄卸載至文字或 CSV 格式。資料會以十六進位形式卸載。您無法卸載包含 FIXEDWIDTH 選項的 VARBYTE 資料。不支援 UNLOAD CSV 的 ADDQUOTES 選項。VARBYTE 資料欄不能是 PARTITIONED BY 資料欄。

FORMAT AS PARQUET 子句

使用 FORMAT AS PARQUET 時，請注意這些考量：

- 卸載至 Parquet 不會使用檔案層級壓縮。每個資料列群組都會使用 SNAPPY 壓縮。
- 如果未指定 MAXFILESIZE，則預設的檔案大小上限為 6.2 GB。您可以使用 MAXFILESIZE 來指定 5 MB – 6.2 GB 的檔案大小。實際檔案大小會在寫入檔案時估算，因此可能不會與您指定的數字完全相等。

  為了最大化掃描效能，Amazon Redshift 會嘗試建立包含 32-MB 相等大小資料列群組的 Parquet 檔案。您指定的 MAXFILESIZE 會自動四捨五入至最近的 32 MB 倍數。例如，如果您指定 MAXFILESIZE 為 200 MB，則每個卸載的 Parquet 檔案大約會是 192 MB (32 MB 資料列群組 x 6 = 192 MB)。
- 如果資料行使用 TIMESTAMPTZ 資料格式，則只會卸載時間戳記的值。不會卸載時區資訊。
- 請不要指定以底線 (_) 或句號 (.) 字元開頭的檔案名稱字首。Redshift Spectrum 會將以這些字元開頭的檔案視為隱藏檔案並進行忽略。

PARTITION BY 子句

使用 PARTITION BY 時，請注意這些考量：

- 分割區資料行不會包含在輸出檔案中。
- 請務必在用於 UNLOAD 陳述式的 SELECT 查詢中包含分割區資料行。您可以在 UNLOAD 命令中指定任何數量的分割區資料行。但是，其中一個限制是檔案內至少應包含一個非分割區資料行。
- 如果分割區索引鍵的值為 Null，Amazon Redshift 會自動將該資料卸載至稱為 `partition_column=__HIVE_DEFAULT_PARTITION__` 的預設分割區。
- UNLOAD 命令不會對外部目錄進行任何呼叫。如要註冊您新的分割區，使其成為您現有外部資料表的一部分，請使用單獨的 ALTER TABLE … ADD PARTITION … 命令。或者，您可以執行 CREATE EXTERNAL TABLE 命令來將已卸載的資料註冊為新的外部資料表。您也可以使用 AWS Glue 爬行者程式來填入資料目錄。如需詳細資訊，請參閱《AWS Glue 開發人員指南》中的定義爬蟲程式。
- 如果您使用 MANIFEST 選項，Amazon Redshift 只會在根 Amazon S3 資料夾中產生一個資訊清單檔案。
- 您可以用來做為分割區索引鍵的資料欄資料類型為 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、BOOLEAN、CHAR、VARCHAR、DATE 和 TIMESTAMP。

使用 ASSUMEROLE 權限授予 (IAM) 角色存取權以進行 UNLOAD 操作

若要讓特定使用者和群組存取 IAM 角色以進行 UNLOAD 操作，超級使用者可以將 IAM 角色的 ASSUMEROLE 權限授予使用者和群組。如需相關資訊，請參閱GRANT。

UNLOAD 不支援 Amazon S3 Access Points 別名

您無法搭配 UNLOAD 命令使用 Amazon S3 Access Points 別名。

範例

如需顯示如何使用 UNLOAD 命令的範例，請參閱 UNLOAD 範例。

UNLOAD 範例

這些範例會示範 UNLOAD 命令的各種參數。許多範例都會使用 TICKIT 範例資料。如需詳細資訊，請參閱 範本資料庫。

> **Note**
>
> 這些範例包含換行以方便閱讀。請勿在 credentials-args 字串中包含換行或空格。

將 VENUE 卸載至縱線分隔檔案 (預設分隔符號)

下列範例會卸載 VENUE 資料表並將資料寫入 s3://mybucket/unload/：

```
unload ('select * from venue')
to 's3://mybucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

根據預設，UNLOAD 會在每個分割中寫入一個或多個檔案。假設有一個雙節點叢集，且每個節點有兩個分割，則上方範例會在 mybucket 中建立這些檔案：

```
unload/0000_part_00
unload/0001_part_00
unload/0002_part_00
unload/0003_part_00
```

為了能更清楚區分輸出檔案，您可以在位置中包含字首。下列範例會卸載 VENUE 資料表並將資料寫入 s3://mybucket/unload/venue_pipe_：

```
unload ('select * from venue')
to 's3://mybucket/unload/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

結果會是 unload 資料夾中的這四個檔案，同樣假設有四個分割。

```
venue_pipe_0000_part_00
venue_pipe_0001_part_00
venue_pipe_0002_part_00
venue_pipe_0003_part_00
```

將 LINEITEM 資料表卸載至已分割的 Parquet 檔案

以下範例會使用以 l_shipdate 資料行分割的 Parquet 格式卸載 LINEITEM 資料表。

```
unload ('select * from lineitem')
to 's3://mybucket/lineitem/'
```

```
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
PARQUET
PARTITION BY (l_shipdate);
```

假設有四個分割，則結果產生的 Parquet 檔案會以動態方式分成不同的資料夾。

```
s3://mybucket/lineitem/l_shipdate=1992-01-02/0000_part_00.parquet
                                             0001_part_00.parquet
                                             0002_part_00.parquet
                                             0003_part_00.parquet
s3://mybucket/lineitem/l_shipdate=1992-01-03/0000_part_00.parquet
                                             0001_part_00.parquet
                                             0002_part_00.parquet
                                             0003_part_00.parquet
s3://mybucket/lineitem/l_shipdate=1992-01-04/0000_part_00.parquet
                                             0001_part_00.parquet
                                             0002_part_00.parquet
                                             0003_part_00.parquet
...
```

Note

在某些情況下，UNLOAD 命令會使用 INCLUDE 選項，如下列 SQL 陳述式中所示。

```
unload ('select * from lineitem')
to 's3://mybucket/lineitem/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
PARQUET
PARTITION BY (l_shipdate) INCLUDE;
```

在這些情況下，`l_shipdate` 資料欄也在 Parquet 檔案的資料中。否則，`l_shipdate` 資料欄資料不在 Parquet 檔案中。

將 VENUE 資料表卸載到 JSON 檔案中

下列範例會卸載 VENUE 資料表，並以 JSON 格式寫入資料至 `s3://mybucket/unload/`。

```
unload ('select * from venue')
to 's3://mybucket/unload/'
```

```
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
JSON;
```

以下是 VENUE 資料表的範例資料列。

```
venueid | venuename                  | venuecity       | venuestate | venueseats
--------+----------------------------+-----------------+------------+-----------
      1 | Pinewood Racetrack         | Akron           | OH         | 0
      2 | Columbus "Crew" Stadium    | Columbus        | OH         | 0
      4 | Community, Ballpark, Arena | Kansas City     | KS         | 0
```

卸載到 JSON 後，檔案的格式類似於以下內容。

```
{"venueid":1,"venuename":"Pinewood
 Racetrack","venuecity":"Akron","venuestate":"OH","venueseats":0}
{"venueid":2,"venuename":"Columbus \"Crew\" Stadium
 ","venuecity":"Columbus","venuestate":"OH","venueseats":0}
{"venueid":4,"venuename":"Community, Ballpark, Arena","venuecity":"Kansas
 City","venuestate":"KS","venueseats":0}
```

將 VENUE 卸載至 CSV 檔案

下列範例會卸載 VENUE 資料表，並以 CSV 格式寫入資料至 s3://mybucket/unload/。

```
unload ('select * from venue')
to 's3://mybucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
CSV;
```

假設 VENUE 資料表包含下列資料列。

```
venueid | venuename                  | venuecity       | venuestate | venueseats
--------+----------------------------+-----------------+------------+-----------
      1 | Pinewood Racetrack         | Akron           | OH         | 0
      2 | Columbus "Crew" Stadium    | Columbus        | OH         | 0
      4 | Community, Ballpark, Arena | Kansas City     | KS         | 0
```

卸載檔案看起來類似下列。

```
1,Pinewood Racetrack,Akron,OH,0
```

```
2,"Columbus ""Crew"" Stadium",Columbus,OH,0
4,"Community, Ballpark, Arena",Kansas City,KS,0
```

使用分隔符號將 VENUE 卸載到 CSV 檔案

下列範例會卸載 VENUE 資料表，並使用縱線字元 (|) 做為分隔符號，寫入 CSV 格式的資料。卸載的檔案會寫入到 s3://mybucket/unload/。此範例中的 VENUE 資料表在第一列 (Pinewood Race|track) 的值中包含縱線字元。它這樣做是為了表明結果中的值用雙引號括住。雙引號會逸出雙引號，而整個欄位會用雙引號括住。

```
unload ('select * from venue')
to 's3://mybucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
CSV DELIMITER AS '|';
```

假設 VENUE 資料表包含下列資料列。

```
venueid | venuename                  | venuecity       | venuestate | venueseats
--------+----------------------------+-----------------+------------+-------------
      1 | Pinewood Race|track        | Akron           | OH         | 0
      2 | Columbus "Crew" Stadium    | Columbus        | OH         | 0
      4 | Community, Ballpark, Arena | Kansas City     | KS         | 0
```

卸載檔案看起來類似下列。

```
1|"Pinewood Race|track"|Akron|OH|0
2|"Columbus ""Crew"" Stadium"|Columbus|OH|0
4|Community, Ballpark, Arena|Kansas City|KS|0
```

使用資訊清單檔案卸載 VENUE

若要建立資訊清單檔案，請包括 MANIFEST 選項。下列範例會卸載 VENUE 資料表，並將資訊清單檔案與資料檔案一起寫入 s3://mybucket/venue_pipe_ :

> **Important**
>
> 若您使用 MANIFEST 選項卸載檔案，則應在載入檔案時使用 MANIFEST 選項搭配 COPY 命令。若您使用相同的字首載入檔案且未指定 MANIFEST 選項，則 COPY 會失敗，因為它會假設資訊清單檔案是資料檔案。

```
unload ('select * from venue')
to 's3://mybucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/
MyRedshiftRole'
manifest;
```

結果會是這五個檔案：

```
s3://mybucket/venue_pipe_0000_part_00
s3://mybucket/venue_pipe_0001_part_00
s3://mybucket/venue_pipe_0002_part_00
s3://mybucket/venue_pipe_0003_part_00
s3://mybucket/venue_pipe_manifest
```

以下示範資訊清單檔案的內容。

```
{
  "entries": [
    {"url":"s3://mybucket/tickit/venue_0000_part_00"},
    {"url":"s3://mybucket/tickit/venue_0001_part_00"},
    {"url":"s3://mybucket/tickit/venue_0002_part_00"},
    {"url":"s3://mybucket/tickit/venue_0003_part_00"}
  ]
}
```

使用 MANIFEST VERBOSE 卸載 VENUE

指定 MANIFEST VERBOSE 選項時，資訊清單檔案會包含下列區段：

- `entries` 區段會列出每個檔案的 Amazon S3 路徑、檔案大小和資料列計數。
- `schema` 區段會列出每個資料欄的資料欄名稱、資料類型和維度。
- `meta` 區段會顯示所有檔案的檔案大小總計和資料列計數。

下列範例會使用 MANIFEST VERBOSE 選項來卸載 VENUE 資料表。

```
unload ('select * from venue')
to 's3://mybucket/unload_venue_folder/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest verbose;
```

以下示範資訊清單檔案的內容。

```
{
  "entries": [
    {"url":"s3://mybucket/venue_pipe_0000_part_00", "meta": { "content_length": 32295,
 "record_count": 10 }},
    {"url":"s3://mybucket/venue_pipe_0001_part_00", "meta": { "content_length": 32771,
 "record_count": 20 }},
    {"url":"s3://mybucket/venue_pipe_0002_part_00", "meta": { "content_length": 32302,
 "record_count": 10 }},
    {"url":"s3://mybucket/venue_pipe_0003_part_00", "meta": { "content_length": 31810,
 "record_count": 15 }}
  ],
  "schema": {
    "elements": [
      {"name": "venueid", "type": { "base": "integer" }},
      {"name": "venuename", "type": { "base": "character varying", 25 }},
      {"name": "venuecity", "type": { "base": "character varying", 25 }},
      {"name": "venuestate", "type": { "base": "character varying", 25 }},
      {"name": "venueseats", "type": { "base": "character varying", 25 }}
    ]
  },
  "meta": {
    "content_length": 129178,
    "record_count": 55
  },
  "author": {
    "name": "Amazon Redshift",
    "version": "1.0.0"
  }
}
```

使用標題卸載 VENUE

以下範例會使用標題列卸載 VENUE。

```
unload ('select * from venue where venueseats > 75000')
to 's3://mybucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
header
parallel off;
```

以下示範資訊具有標題列的輸出檔案的內容。

```
venueid|venuename|venuecity|venuestate|venueseats
```

```
6|New York Giants Stadium|East Rutherford|NJ|80242
78|INVESCO Field|Denver|CO|76125
83|FedExField|Landover|MD|91704
79|Arrowhead Stadium|Kansas City|MO|79451
```

將 VENUE 卸載至更小的檔案

根據預設，檔案大小的上限為 6.2 GB。如果卸載資料大於 6.2 GB，則 UNLOAD 會為每個 6.2 GB 資料區段建立一個新檔案。若要建立更小的檔案，請包括 MAXFILESIZE 參數。假設先前範例中的資料大小為 20 GB，則下列 UNLOAD 命令會建立 20 個檔案，每個檔案的大小為 1 GB。

```
unload ('select * from venue')
to 's3://mybucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
maxfilesize 1 gb;
```

依序卸載 VENUE

若要依序卸載，請指定 PARALLEL OFF。如此 UNLOAD 就會一次寫入一個檔案，直到每個檔案達到 6.2 GB 上限為止。

下列範例會卸載 VENUE 資料表並依序將資料寫入 s3://mybucket/unload/。

```
unload ('select * from venue')
to 's3://mybucket/unload/venue_serial_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off;
```

結果會產生一個名為 venue_serial_000 的檔案。

如果卸載資料大於 6.2 GB，則 UNLOAD 會為每個 6.2 GB 資料區段建立一個新檔案。下列範例會卸載 LINEORDER 資料表並依序將資料寫入 s3://mybucket/unload/。

```
unload ('select * from lineorder')
to 's3://mybucket/unload/lineorder_serial_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off gzip;
```

結果會是下方的一系列檔案。

```
lineorder_serial_0000.gz
```

```
lineorder_serial_0001.gz
lineorder_serial_0002.gz
lineorder_serial_0003.gz
```

為了能更清楚區分輸出檔案，您可以在位置中包含字首。下列範例會卸載 VENUE 資料表並將資料寫入 s3://mybucket/venue_pipe_：

```
unload ('select * from venue')
to 's3://mybucket/unload/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

結果會是 unload 資料夾中的這四個檔案，同樣假設有四個分割。

```
venue_pipe_0000_part_00
venue_pipe_0001_part_00
venue_pipe_0002_part_00
venue_pipe_0003_part_00
```

從卸載檔案載入 VENUE

若要從一組卸載檔案載入資料表，只要使用 COPY 命令將程序反向即可。下列範例會建立 LOADVENUE 這個新資料表，並從先前範例中建立的資料檔案載入資料表。

```
create table loadvenue (like venue);

copy loadvenue from 's3://mybucket/venue_pipe_' iam_role
 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

若您使用 MANIFEST 選項在卸載檔案中建立資訊清單檔案，則您可以使用相同的資訊清單檔案載入資料。您可以使用 COPY 命令搭配 MANIFEST 選項來執行此動作。下列範例會使用資訊清單檔案載入資料。

```
copy loadvenue
from 's3://mybucket/venue_pipe_manifest' iam_role 'arn:aws:iam::0123456789012:role/
MyRedshiftRole'
manifest;
```

將 VENUE 卸載至加密檔案

下列範例會使用 AWS KMS 金鑰將 VENUE 資料表卸載至一組加密檔案。若您使用 ENCRYPTED 選項指定資訊清單檔案，則資訊清單檔案也會加密。如需詳細資訊，請參閱 卸載加密的資料檔案。

```
unload ('select * from venue')
to 's3://mybucket/venue_encrypt_kms'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
kms_key_id '1234abcd-12ab-34cd-56ef-1234567890ab'
manifest
encrypted;
```

下列範例會使用根對稱金鑰將 VENUE 資料表卸載至一組加密檔案。

```
unload ('select * from venue')
to 's3://mybucket/venue_encrypt_cmk'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key 'EXAMPLEMASTERKEYtkbjk/OpCwtYSx/M4/t7DMCDIK722'
encrypted;
```

從加密檔案載入 VENUE

若要從使用 UNLOAD 搭配 ENCRYPT 選項建立的一組檔案中載入資料表，請使用 COPY 命令執行反向程序。利用該命令，使用 ENCRYPTED 選項並指定用於 UNLOAD 命令的相同根對稱金鑰。下列範例會從先前範例中建立的加密資料檔案載入 LOADVENUE 資料表。

```
create table loadvenue (like venue);

copy loadvenue
from 's3://mybucket/venue_encrypt_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key 'EXAMPLEMASTERKEYtkbjk/OpCwtYSx/M4/t7DMCDIK722'
manifest
encrypted;
```

將 VENUE 資料卸載至 Tab 分隔檔案

```
unload ('select venueid, venuename, venueseats from venue')
to 's3://mybucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter as '\t';
```

輸出資料檔案看起來像這樣：

```
1 Toyota Park Bridgeview IL 0
2 Columbus Crew Stadium Columbus OH 0
```

```
3 RFK Stadium Washington DC 0
4 CommunityAmerica Ballpark Kansas City KS 0
5 Gillette Stadium Foxborough MA 68756
...
```

將 VENUE 卸載至固定寬度資料檔案

```
unload ('select * from venue')
to 's3://mybucket/venue_fw_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth as 'venueid:3,venuename:39,venuecity:16,venuestate:2,venueseats:6';
```

輸出資料檔案看起來會像下面這樣。

```
1  Toyota Park              Bridgeview  IL0
2  Columbus Crew Stadium    Columbus    OH0
3  RFK Stadium              Washington  DC0
4  CommunityAmerica BallparkKansas City KS0
5  Gillette Stadium         Foxborough  MA68756
...
```

將 VENUE 卸載至一組 Tab 分隔的 GZIP 壓縮檔案

```
unload ('select * from venue')
to 's3://mybucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter as '\t'
gzip;
```

將 VENUE 卸載到一個 GZIP 壓縮的文字檔案

```
unload ('select * from venue')
to 's3://mybucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
extension 'txt.gz'
gzip;
```

卸載包含分隔符號的資料

此範例會使用 ADDQUOTES 選項卸載逗號分隔的資料，其中部分實際資料欄位包含逗號。

首先，請建立包含引號的資料表。

```
create table location (id int, location char(64));

insert into location values (1,'Phoenix, AZ'),(2,'San Diego, CA'),(3,'Chicago, IL');
```

接著使用 ADDQUOTES 選項卸載資料。

```
unload ('select id, location from location')
to 's3://mybucket/location_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter ',' addquotes;
```

卸載的資料檔案看起來像這樣：

```
1,"Phoenix, AZ"
2,"San Diego, CA"
3,"Chicago, IL"
...
```

卸載聯結查詢的結果

以下範例會卸載包含視窗函數之聯結查詢的結果。

```
unload ('select venuecity, venuestate, caldate, pricepaid,
sum(pricepaid) over(partition by venuecity, venuestate
order by caldate rows between 3 preceding and 3 following) as winsum
from sales join date on sales.dateid=date.dateid
join event on event.eventid=sales.eventid
join venue on event.venueid=venue.venueid
order by 1,2')
to 's3://mybucket/tickit/winsum'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

輸出檔案看起來像這樣：

```
Atlanta|GA|2008-01-04|363.00|1362.00
Atlanta|GA|2008-01-05|233.00|2030.00
Atlanta|GA|2008-01-06|310.00|3135.00
Atlanta|GA|2008-01-08|166.00|8338.00
Atlanta|GA|2008-01-11|268.00|7630.00
...
```

使用 NULL AS 卸載

根據預設，UNLOAD 會將 null 值做為空字串輸出。下列範例說明如何使用 NULL AS 將文字字串替換為 null。

我們將在這些範例的 VENUE 資料表中加入一些 null 值。

```
update venue set venuestate = NULL
where venuecity = 'Cleveland';
```

從其中 VENUESTATE 是 null 的 VENUE 選取，以確認資料欄包含 NULL。

```
select * from venue where venuestate is null;

 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
      22 | Quicken Loans Arena      | Cleveland |            |          0
     101 | Progressive Field        | Cleveland |            |      43345
      72 | Cleveland Browns Stadium | Cleveland |            |      73200
```

現在使用 NULL AS 選項對 VENUE 資料表執行 UNLOAD，將 null 值取代為字元字串 'fred'。

```
unload ('select * from venue')
to 's3://mybucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
null as 'fred';
```

下列來自卸載檔案的範例顯示，null 值已取代為 fred。另外發現，VENUESEATS 的某些值也是 null，並且已取代為 fred。即使 VENUESEATS 的資料類型為整數，UNLOAD 仍會在卸載檔案中將值轉換成文字，然後 COPY 會再將它們還原為整數。若您要卸載至固定寬度的檔案，則 NULL AS 字串不得大於欄位寬度。

```
248|Charles Playhouse|Boston|MA|0
251|Paris Hotel|Las Vegas|NV|fred
258|Tropicana Hotel|Las Vegas|NV|fred
300|Kennedy Center Opera House|Washington|DC|0
306|Lyric Opera House|Baltimore|MD|0
308|Metropolitan Opera|New York City|NY|0
  5|Gillette Stadium|Foxborough|MA|5
 22|Quicken Loans Arena|Cleveland|fred|0
```

```
101|Progressive Field|Cleveland|fred|43345
...
```

若要從卸載檔案載入資料表，請使用 COPY 命令搭配相同的 NULL AS 選項。

> **Note**
>
> 如果嘗試將 null 載入已定義為 NOT NULL 的欄，COPY 命令會失敗。

```
create table loadvenuenulls (like venue);

copy loadvenuenulls from 's3://mybucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
null as 'fred';
```

若要確認資料欄包含 null，而不只是空字串，請從 LOADVENUENULLS 選取並篩選出 null。

```
select * from loadvenuenulls where venuestate is null or venueseats is null;

 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
      72 | Cleveland Browns Stadium | Cleveland |            |      73200
     253 | Mirage Hotel             | Las Vegas | NV         |
     255 | Venetian Hotel           | Las Vegas | NV         |
      22 | Quicken Loans Arena      | Cleveland |            |          0
     101 | Progressive Field        | Cleveland |            |      43345
     251 | Paris Hotel              | Las Vegas | NV         |

...
```

您可以使用預設的 NULL AS 行為 UNLOAD 包含 null 的資料表，然後使用預設的 NULL AS 行為將資料 COPY 回資料表中；不過，目標資料表中任何非數值欄位都會包含空字串，而非 null。根據預設，UNLOAD 會將 null 轉換成空字串 (空格或零長度)。COPY 會針對數值資料欄將空字串轉換成 NULL，但是會將空字串插入非數值資料欄。下列範例說明如何使用預設的 NULL AS 行為來執行 UNLOAD，後面接著 COPY。

```
unload ('select * from venue')
to 's3://mybucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' allowoverwrite;
```

```
truncate loadvenuenulls;
copy loadvenuenulls from 's3://mybucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

在此情況下，當您篩選出 null 時，只會有 VENUESEATS 包含 null 的資料列。VENUESTATE 在資料表 (VENUE) 中包含 null，而 VENUESTATE 在目標資料表 (LOADVENUENULLS) 中則包含空字串。

```
select * from loadvenuenulls where venuestate is null or venueseats is null;

 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
     253 | Mirage Hotel             | Las Vegas | NV         |
     255 | Venetian Hotel           | Las Vegas | NV         |
     251 | Paris Hotel              | Las Vegas | NV         |
...
```

若要在非數值資料欄中將空字串載入為 NULL，請包含 EMPTYASNULL 或 BLANKSASNULL 選項。也可以兩者都使用。

```
unload ('select * from venue')
to 's3://mybucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' allowoverwrite;

truncate loadvenuenulls;
copy loadvenuenulls from 's3://mybucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' EMPTYASNULL;
```

若要確認資料欄包含 NULL，而不只是空格或空字串，請從 LOADVENUENULLS 選取並篩選出 null。

```
select * from loadvenuenulls where venuestate is null or venueseats is null;

 venueid |        venuename         | venuecity | venuestate | venueseats
---------+--------------------------+-----------+------------+------------
      72 | Cleveland Browns Stadium | Cleveland |            |      73200
     253 | Mirage Hotel             | Las Vegas | NV         |
     255 | Venetian Hotel           | Las Vegas | NV         |
      22 | Quicken Loans Arena      | Cleveland |            |          0
     101 | Progressive Field        | Cleveland |            |      43345
     251 | Paris Hotel              | Las Vegas | NV         |
     ...
```

使用 ALLOWOVERWRITE 參數卸載

根據預設，UNLOAD 不會覆寫目的地儲存貯體中的現有檔案。例如，如果您執行相同的 UNLOAD 陳述式兩次，而未修改目的地儲存貯體中的檔案，則第二個 UNLOAD 會失敗。為覆寫現有檔案 (包括資訊清單檔案)，指定 ALLOWOVERWRITE 選項。

```
unload ('select * from venue')
to 's3://mybucket/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest allowoverwrite;
```

使用 PARALLEL 和 MANIFEST 參數卸載 EVENT 資料表

您可以以平行方式 UNLOAD 資料表並產生清單檔案。Amazon S3 資料檔案都是在相同層級建立的，名稱字尾為模式 `0000_part_00`。清單檔案與資料檔案位於相同的資料夾層級，並加上字尾 `manifest`。下面的 SQL 會卸載 EVENT 資料表，並季戀具有基本名稱 `parallel` 的文件

```
unload ('select * from mytickit1.event')
to 's3://my-s3-bucket-name/parallel'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
parallel on
manifest;
```

Amazon S3 檔案清單類似以下內容。

```
 Name                        Last modified                          Size
 parallel0000_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 52.1 KB
 parallel0001_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 53.4 KB
 parallel0002_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 52.1 KB
 parallel0003_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 51.1 KB
 parallel0004_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 54.6 KB
 parallel0005_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 53.4 KB
 parallel0006_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 54.1 KB
 parallel0007_part_00 -   August 2, 2023, 14:54:39 (UTC-07:00) 55.9 KB
 parallelmanifest       -   August 2, 2023, 14:54:39 (UTC-07:00) 886.0 B
```

`parallelmanifest` 檔案內容類似以下內容。

```
{
  "entries": [
    {"url":"s3://my-s3-bucket-name/parallel0000_part_00", "meta": { "content_length":
 53316 }},
    {"url":"s3://my-s3-bucket-name/parallel0001_part_00", "meta": { "content_length":
 54704 }},
    {"url":"s3://my-s3-bucket-name/parallel0002_part_00", "meta": { "content_length":
 53326 }},
    {"url":"s3://my-s3-bucket-name/parallel0003_part_00", "meta": { "content_length":
 52356 }},
    {"url":"s3://my-s3-bucket-name/parallel0004_part_00", "meta": { "content_length":
 55933 }},
    {"url":"s3://my-s3-bucket-name/parallel0005_part_00", "meta": { "content_length":
 54648 }},
    {"url":"s3://my-s3-bucket-name/parallel0006_part_00", "meta": { "content_length":
 55436 }},
    {"url":"s3://my-s3-bucket-name/parallel0007_part_00", "meta": { "content_length":
 57272 }}
  ]
}
```

使用 PARALLEL OFF 和 MANIFEST 參數卸載 EVENT 資料表

您可以依序 (PARALLEL OFF) UNLOAD 資料表並產生清單檔案。Amazon S3 資料檔案都是在相同層級建立的，名稱字尾為模式 0000。清單檔案與資料檔案位於相同的資料夾層級，並加上字尾 manifest。

```
unload ('select * from mytickit1.event')
to 's3://my-s3-bucket-name/serial'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
parallel off
manifest;
```

Amazon S3 檔案清單類似以下內容。

```
 Name                          Last modified                          Size
 serial0000              -     August 2, 2023, 15:54:39 (UTC-07:00) 426.7 KB
 serialmanifest          -     August 2, 2023, 15:54:39 (UTC-07:00) 120.0 B
```

serialmanifest 檔案內容類似以下內容。

```
{
  "entries": [
    {"url":"s3://my-s3-bucket-name/serial000", "meta": { "content_length": 436991 }}
  ]
}
```

使用 PARTITION BY 和 MANIFEST 參數卸載 EVENT 資料表

您可以依分割區 UNLOAD 資料表並產生清單檔案。Amazon S3 中會建立一個新資料夾，其中包含子分割區資料夾，而子資料夾中的資料檔案名稱模式類似於 0000_par_00。清單檔案與名稱為 manifest 的子資料夾位於相同的資料夾層級。

```
unload ('select * from mytickit1.event')
to 's3://my-s3-bucket-name/partition'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
partition by (eventname)
manifest;
```

Amazon S3 檔案清單類似以下內容。

```
 Name                   Type     Last modified                     Size

 partition            Folder
```

在 partition 資料夾中的是具有分割區名稱和清單檔案的子文件夾。下面顯示的是資料夾 partition 中資料夾清單的底端，類似以下內容。

```
 Name                   Type      Last modified                      Size

 ...
 eventname=Zucchero/    Folder
 eventname=Zumanity/    Folder
 eventname=ZZ Top/      Folder
 manifest               -     August 2, 2023, 15:54:39 (UTC-07:00) 467.6 KB
```

eventname=Zucchero/ 資料夾中的資料檔案類似於以下內容。

```
 Name               Last modified                        Size
 0000_part_00 -   August 2, 2023, 15:59:19 (UTC-07:00) 70.0 B
 0001_part_00 -   August 2, 2023, 15:59:16 (UTC-07:00) 106.0 B
 0002_part_00 -   August 2, 2023, 15:59:15 (UTC-07:00) 70.0 B
 0004_part_00 -   August 2, 2023, 15:59:17 (UTC-07:00) 141.0 B
 0006_part_00 -   August 2, 2023, 15:59:16 (UTC-07:00) 35.0 B
 0007_part_00 -   August 2, 2023, 15:59:19 (UTC-07:00) 108.0 B
```

manifest 檔案內容底部類似以下內容。

```
{
  "entries": [
    ...
    {"url":"s3://my-s3-bucket-name/partition/eventname=Zucchero/0007_part_00", "meta":
 { "content_length": 108 }},
    {"url":"s3://my-s3-bucket-name/partition/eventname=Zumanity/0007_part_00", "meta":
 { "content_length": 72 }}
  ]
}
```

使用 MAXFILESIZE、ROWGROUPSIZE 和 MANIFEST 參數卸載 EVENT 資料表

您可以以平行方式 UNLOAD 資料表並產生清單檔案。Amazon S3 資料檔案都是在相同層級建立的，名稱字尾為模式 0000_part_00。產生的 Parquet 資料檔案會限制為 256 MB，資料列群組大小為 128 MB。清單檔案與資料檔案位於相同的資料夾層級，並加上字尾 manifest。

```
unload ('select * from mytickit1.event')
to 's3://my-s3-bucket-name/eventsize'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
maxfilesize 256 MB
rowgroupsize 128 MB
parallel on
parquet
manifest;
```

Amazon S3 檔案清單類似以下內容。

```
 Name                              Type      Last modified                        Size
 eventsize0000_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.5 KB
```

```
 eventsize0001_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.8 KB
 eventsize0002_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.4 KB
 eventsize0003_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.0 KB
 eventsize0004_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 25.3 KB
 eventsize0005_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 24.8 KB
 eventsize0006_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 25.0 KB
 eventsize0007_part_00.parquet parquet August 2, 2023, 17:35:21 (UTC-07:00) 25.6 KB
 eventsizemanifest                 -       August 2, 2023, 17:35:21 (UTC-07:00) 958.0 B
```

`eventsizemanifest` 檔案內容類似以下內容。

```
{
  "entries": [
    {"url":"s3://my-s3-bucket-name/eventsize0000_part_00.parquet", "meta":
 { "content_length": 25130 }},
    {"url":"s3://my-s3-bucket-name/eventsize0001_part_00.parquet", "meta":
 { "content_length": 25428 }},
    {"url":"s3://my-s3-bucket-name/eventsize0002_part_00.parquet", "meta":
 { "content_length": 25025 }},
    {"url":"s3://my-s3-bucket-name/eventsize0003_part_00.parquet", "meta":
 { "content_length": 24554 }},
    {"url":"s3://my-s3-bucket-name/eventsize0004_part_00.parquet", "meta":
 { "content_length": 25918 }},
    {"url":"s3://my-s3-bucket-name/eventsize0005_part_00.parquet", "meta":
 { "content_length": 25362 }},
    {"url":"s3://my-s3-bucket-name/eventsize0006_part_00.parquet", "meta":
 { "content_length": 25647 }},
    {"url":"s3://my-s3-bucket-name/eventsize0007_part_00.parquet", "meta":
 { "content_length": 26256 }}
  ]
}
```

UPDATE

主題

在條件滿足時，更新一個或多個資料表資料欄中的值。

> **Note**
>
> 單一 SQL 陳述式的大小上限為 16 MB。

語法

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
            UPDATE table_name [ [ AS ] alias ] SET column = { expression | DEFAULT }
 [,...]

[ FROM fromlist ]
[ WHERE condition ]
```

參數

WITH 子句

指定一個或多個 common-table-expressions 的選用子句。請參閱WITH 子句。

table_name

暫時性或持久性資料表。只有資料表的擁有者，或具有資料表 UPDATE 權限的使用者可以更新資料列。如果您在表達式或條件中使用 FROM 子句或從資料表選取，則必須具有這些資料表的 SELECT 權限。您無法在這裡為資料表提供別名；不過，您可以在 FROM 子句中指定別名。

> **Note**
>
> Amazon Redshift Spectrum 外部資料表處於唯讀狀態。您無法 UPDATE 外部資料表。

別名

目標資料表的暫時替代名稱。別名是選擇性項目。AS 關鍵字一律為選用。

SET column =

您要修改的一個或多個資料欄。未列出的資料欄會保留其目前值。不要在目標資料欄的規格中包含資料表名稱。例如，`UPDATE tab SET tab.col = 1` 無效。

運算式

此表達式會為所指定資料欄定義新值。

DEFAULT

使用在 CREATE TABLE 陳述式中為資料欄指派的預設值更新資料欄。

FROM tablelist

您可以透過參考其他資料表中的資訊更新資料表。請在 FROM 子句中列出這些其他資料表，或在 WHERE 條件中使用子查詢。FROM 子句中列出的資料表可以有別名。如果您需要在清單中包含 UPDATE 陳述式的目標資料表，請使用別名。

WHERE condition

此選用子句會限制僅更新符合條件的資料列。當條件傳回 `true` 時，指定的 SET 欄就會更新。條件可以是資料欄上簡單的述詞，或是根據子查詢結果的條件。

您可以在子查詢中為任何資料表命名，包括 UPDATE 的目標資料表在內。

使用須知

在更新資料表中的大量資料列之後：

- 清空資料表以回收儲存空間和重新排序資料列。
- 分析資料表以更新查詢規劃器的統計資訊。

UPDATE 陳述式的 FROM 子句中不支援左、右和完整外部聯結；它們會傳回下列錯誤：

```
ERROR: Target table must be part of an equijoin predicate
```

若您需要指定外部聯結，可在 UPDATE 陳述式的 WHERE 子句中使用子查詢。

若您的 UPDATE 陳述式需要自我聯結至目標資料表，則您需要指定聯結條件以及 WHERE 子句條件來限定更新操作的資料列。大致而言，當目標資料表與其本身或其他資料表聯結時，最佳實務會是使用子查詢清楚區分聯結條件和限定更新資料列的條件。

當組態參數 `error_on_nondeterministic_update` 設定為 true 時，每列具有多個相符項目的 UPDATE 查詢會擲回錯誤。如需詳細資訊，請參閱 error_on_nondeterministic_update。

您可以更新 GENERATED BY DEFAULT AS IDENTITY 資料欄。定義為 GENERATED BY DEFAULT AS IDENTITY 的資料欄可以利用您提供的值進行更新。如需詳細資訊，請參閱 GENERATED BY DEFAULT AS IDENTITY。

UPDATE 陳述式的範例

如需下列範例中所用資料表的相關資訊，請參閱 範本資料庫。

TICKIT 資料庫中的 CATEGORY 資料表包含以下資料列：

```
+-------+----------+-----------+--------------------------------------------+
| catid | catgroup |  catname  |                  catdesc                   |
+-------+----------+-----------+--------------------------------------------+
| 5     | Sports   | MLS       | Major League Soccer                        |
| 11    | Concerts | Classical | All symphony, concerto, and choir concerts |
| 1     | Sports   | MLB       | Major League Baseball                      |
| 6     | Shows    | Musicals  | Musical theatre                            |
| 3     | Sports   | NFL       | National Football League                   |
| 8     | Shows    | Opera     | All opera and light opera                  |
| 2     | Sports   | NHL       | National Hockey League                     |
| 9     | Concerts | Pop       | All rock and pop music concerts            |
| 4     | Sports   | NBA       | National Basketball Association            |
| 7     | Shows    | Plays     | All non-musical theatre                    |
| 10    | Concerts | Jazz      | All jazz singers and bands                 |
+-------+----------+-----------+--------------------------------------------+
```

根據值的範圍更新資料表

根據 CATID 資料欄中的值範圍更新 CATGROUP 資料欄。

```
UPDATE category
SET catgroup='Theatre'
WHERE catid BETWEEN 6 AND 8;

SELECT * FROM category
WHERE catid BETWEEN 6 AND 8;

+-------+----------+----------+--------------------------+
| catid | catgroup | catname  |          catdesc         |
+-------+----------+----------+--------------------------+
| 6     | Theatre  | Musicals | Musical theatre          |
| 7     | Theatre  | Plays    | All non-musical theatre  |
| 8     | Theatre  | Opera    | All opera and light opera |
```

```
+-------+----------+----------+--------------------------+
```

根據目前值更新資料表

根據目前的 CATGROUP 值更新 CATNAME 和 CATDESC 資料欄：

```
UPDATE category
SET catdesc=default, catname='Shows'
WHERE catgroup='Theatre';

SELECT * FROM category
WHERE catname='Shows';

+-------+----------+---------+---------+
| catid | catgroup | catname | catdesc |
+-------+----------+---------+---------+
| 6     | Theatre  | Shows   | NULL    |
| 7     | Theatre  | Shows   | NULL    |
| 8     | Theatre  | Shows   | NULL    |
+-------+----------+---------+---------+)
```

在此情況下，CATDESC 資料欄已設為 null，因為資料表建立時並未定義任何預設值。

執行下列命令，將 CATEGORY 資料表資料重設回原始值：

```
TRUNCATE category;

COPY category
FROM 's3://redshift-downloads/tickit/category_pipe.txt'
DELIMITER '|'
IGNOREHEADER 1
REGION 'us-east-1'
IAM_ROLE default;
```

根據 WHERE 子句子查詢的結果更新資料表

根據 WHERE 子句中子查詢的結果更新 CATEGORY 資料表：

```
UPDATE category
SET catdesc='Broadway Musical'
WHERE category.catid IN
(SELECT category.catid FROM category
JOIN event ON category.catid = event.catid
```

```
JOIN venue ON venue.venueid = event.venueid
JOIN sales ON sales.eventid = event.eventid
WHERE venuecity='New York City' AND catname='Musicals');
```

檢視更新的資料表：

```
SELECT * FROM category ORDER BY catid;

+-------+----------+-----------+--------------------------------------------+
| catid | catgroup |  catname  |                  catdesc                   |
+-------+----------+-----------+--------------------------------------------+
| 2     | Sports   | NHL       | National Hockey League                     |
| 3     | Sports   | NFL       | National Football League                   |
| 4     | Sports   | NBA       | National Basketball Association            |
| 5     | Sports   | MLS       | Major League Soccer                        |
| 6     | Shows    | Musicals  | Broadway Musical                           |
| 7     | Shows    | Plays     | All non-musical theatre                    |
| 8     | Shows    | Opera     | All opera and light opera                  |
| 9     | Concerts | Pop       | All rock and pop music concerts            |
| 10    | Concerts | Jazz      | All jazz singers and bands                 |
| 11    | Concerts | Classical | All symphony, concerto, and choir concerts |
+-------+----------+-----------+--------------------------------------------+
```

根據 WITH 子句子查詢的結果更新資料表

若要根據使用 WITH 子句之子查詢的結果來更新 CATEGORY 資料表，請使用下列範例。

```
WITH u1 as (SELECT catid FROM event ORDER BY catid DESC LIMIT 1)
UPDATE category SET catid='200' FROM u1 WHERE u1.catid=category.catid;

SELECT * FROM category ORDER BY catid DESC LIMIT 1;

+-------+----------+---------+----------------------------------+
| catid | catgroup | catname |             catdesc              |
+-------+----------+---------+----------------------------------+
| 200   | Concerts | Pop     | All rock and pop music concerts |
+-------+----------+---------+----------------------------------+
```

根據聯結條件的結果更新資料表

根據 EVENT 資料表中相符的 CATID 資料列，更新 CATEGORY 資料表中原有的 11 個資料列：

```
UPDATE category SET catid=100
```

```
FROM event
WHERE event.catid=category.catid;

SELECT * FROM category ORDER BY catid;

+-------+----------+-----------+--------------------------------------------+
| catid | catgroup |  catname  |                  catdesc                   |
+-------+----------+-----------+--------------------------------------------+
| 2     | Sports   | NHL       | National Hockey League                     |
| 3     | Sports   | NFL       | National Football League                   |
| 4     | Sports   | NBA       | National Basketball Association            |
| 5     | Sports   | MLS       | Major League Soccer                        |
| 10    | Concerts | Jazz      | All jazz singers and bands                 |
| 11    | Concerts | Classical | All symphony, concerto, and choir concerts |
| 100   | Concerts | Pop       | All rock and pop music concerts            |
| 100   | Shows    | Plays     | All non-musical theatre                    |
| 100   | Shows    | Opera     | All opera and light opera                  |
| 100   | Shows    | Musicals  | Broadway Musical                           |
+-------+----------+-----------+--------------------------------------------+
```

請注意，EVENT 會在 FROM 子句中列出，而目標資料表的聯結條件會在 WHERE 子句中定義。只有四個資料列符合更新資格。這四個資料列是 CATID 值原本為 6、7、8 和 9 的資料列；EVENT 資料表中只會表示這四種類別：

```
SELECT DISTINCT catid FROM event;

+-------+
| catid |
+-------+
| 6     |
| 7     |
| 8     |
| 9     |
+-------+
```

藉由延伸先前的範例並新增其他條件至 WHERE 子句，更新 CATEGORY 資料表中原有的 11 個資料列。由於 CATGROUP 資料欄的限制，只有一個資料列符合更新資格 (雖然四個資料列都符合聯結資格)。

```
UPDATE category SET catid=100
FROM event
WHERE event.catid=category.catid
```

```
AND catgroup='Concerts';

SELECT * FROM category WHERE catid=100;

+-------+----------+---------+----------------------------------+
| catid | catgroup | catname |             catdesc              |
+-------+----------+---------+----------------------------------+
| 100   | Concerts | Pop     | All rock and pop music concerts |
+-------+----------+---------+----------------------------------+
```

撰寫此範例的替代方法如下：

```
UPDATE category SET catid=100
FROM event JOIN category cat ON event.catid=cat.catid
WHERE cat.catgroup='Concerts';
```

此方法的好處在於，聯結條件會與任何其他限定更新資料列的條件清楚區分。請注意 FROM 子句中的 CATEGORY 資料表使用別名 CAT。

使用 FROM 子句中外部聯結的更新

上面的範例說明了 UPDATE 陳述式的 FROM 子句中指定的內部聯結。下列範例會傳回錯誤，因為 FROM 子句不支援目標資料表的外部聯結：

```
UPDATE category SET catid=100
FROM event LEFT JOIN category cat ON event.catid=cat.catid
WHERE cat.catgroup='Concerts';
ERROR:  Target table must be part of an equijoin predicate
```

若 UPDATE 陳述式需要外部聯結，您可以將外部聯結語法移到子查詢內：

```
UPDATE category SET catid=100
FROM
(SELECT event.catid FROM event LEFT JOIN category cat ON event.catid=cat.catid)
 eventcat
WHERE category.catid=eventcat.catid
AND catgroup='Concerts';
```

更新 SET 子句中另一個資料表的資料欄

若要使用 sales 資料表中的值更新 TICKIT 範例資料庫中的 listing 資料表，請使用下列範例。

```
SELECT listid, numtickets FROM listing WHERE sellerid = 1 ORDER BY 1 ASC LIMIT 5;

+--------+------------+
| listid | numtickets |
+--------+------------+
| 100423 | 4          |
| 108334 | 24         |
| 117150 | 4          |
| 135915 | 20         |
| 205927 | 6          |
+--------+------------+

UPDATE listing
SET numtickets = sales.sellerid
FROM sales
WHERE sales.sellerid = 1 AND listing.sellerid = sales.sellerid;

SELECT listid, numtickets FROM listing WHERE sellerid = 1 ORDER BY 1 ASC LIMIT 5;

+--------+------------+
| listid | numtickets |
+--------+------------+
| 100423 | 1          |
| 108334 | 1          |
| 117150 | 1          |
| 135915 | 1          |
| 205927 | 1          |
+--------+------------+
```

VACUUM

在指定的資料表中或目前資料庫的所有資料表中，重新排序資料列並回收空間。

> **Note**
>
> 只有具有必要資料表權限的使用者才能有效地清除資料表。若 VACUUM 執行時沒有必要的資料表權限，操作仍會成功完成，但不會有任何作用。如需有效執行 VACUUM 的有效資料表權限清單，請參閱下列「所需權限」一節。

Amazon Redshift 會自動排序資料並在背景中執行 VACUUM DELETE。這可以減少執行 VACUUM 命令的需求。如需詳細資訊，請參閱 清空資料表。

根據預設，若資料表中超過 95% 的資料列已排序，則 VACUUM 會略過資料表的排序階段。略過排序階段可大幅改善 VACUUM 的效能。若要變更單一資料表預設的排序或刪除閾值，則在執行 VACUUM 命令時，包含資料表名稱和 TO threshold PERCENT 參數。

使用者可以在清空資料表時加以存取。在清空資料表時您可以執行查詢和寫入操作，但是當資料處理語言 (DML) 命令和清空並行執行時，這兩項可能都需要較長時間。如果您在清空期間執行 UPDATE 和 DELETE 陳述式，系統效能可能會降低。VACUUM DELETE 展示封鎖更新和刪除操作。

Amazon Redshift 會在背景自動執行 DELETE ONLY 清空。使用者執行資料定義語言 (DDL) 操作 (例如 ALTER TABLE) 時，自動清空操作將暫停。

> **Note**
>
> Amazon Redshift VACUUM 命令語法和行為與 PostgreSQL VACUUM 操作有著顯著的差異。例如，Amazon Redshift 中的預設 VACUUM 操作為 VACUUM FULL，其會回收磁碟空間並重新排序所有資料列。相反地，PostgreSQL 中的預設 VACUUM 操作只會回收空間，並將空間提供重複使用。

如需詳細資訊，請參閱 清空資料表。

所需權限

以下是 VACUUM 所需的權限：

- 超級使用者
- 具有 VACUUM 權限的使用者
- 資料表擁有者
- 共用資料表的資料庫擁有者

語法

```
VACUUM [ FULL | SORT ONLY | DELETE ONLY | REINDEX | RECLUSTER ]
[ [ table_name ] [ TO threshold PERCENT ] [ BOOST ] ]
```

參數

FULL

排序指定的資料表 (或目前資料庫中的所有資料表)，並回收先前 UPDATE 和 DELETE 操作將其標記為要進行刪除的資料列所佔用的磁碟空間。VACUUM FULL 是預設值。

完整清空並不會對交錯資料表執行重新建立索引。若要在完整清空之後重新建立交錯資料表的索引，請使用 VACUUM REINDEX 選項。

根據預設，若資料表中至少有 95% 的資料列已排序，則 VACUUM FULL 會略過資料表的排序階段。若 VACUUM 能夠略過排序階段，它就會執行 DELETE ONLY，並且在刪除階段中回收空間，如此至少有 95% 的剩餘資料列不會標記為要進行刪除。

若未達排序閾值 (例如 90% 的資料列已排序)，且 VACUUM 執行了完整排序，則也會執行完整刪除操作，並復原已刪除資料列的全部空間。

您只能變更單一資料表的預設清空閾值。若要變更單一資料表預設的清空閾值，請包含資料表名稱和 TO threshold PERCENT 參數。

SORT ONLY

排序指定的資料表 (或目前資料庫中的所有資料表)，但不回收已刪除資料列釋出的空間。當回收磁碟空間不重要，但重新排序新資料列很重要時，此選項很實用。若未排序的區域未包含大量已刪除資料列，且未跨到整個已排序區域，則 SORT ONLY 清空可縮短清空操作的經過時間。若應用程式沒有磁碟空間限制條件，但須倚賴與保持資料表資料列排序狀態相關聯的查詢最佳化，則可受益於這類清空。

根據預設，若資料表中至少有 95% 的內容已排序，則 VACUUM SORT ONLY 會略過資料表。若要變更單一資料表預設的排序閾值，則在執行 VACUUM 時，包含資料表名稱和 TO threshold PERCENT 參數。

DELETE ONLY

Amazon Redshift 僅在背景執行 DELETE ONLY 清空，因此您很少需要執行 DELETE ONLY 清空。

回收先前 UPDATE 和 DELETE 操作將其標記為要進行刪除的資料列所佔用的 VACUUM DELETE 磁碟空間，並使資料表精簡以釋出耗用的空間。DELETE ONLY 清空操作不會排序資料表的資料。

當回收磁碟空間很重要，但重新排序新資料列不重要時，此選項可縮短清空操作的經過時間。當您的查詢效能已達最佳狀態，而不需要重新排序資料列來最佳化查詢效能時，此選項也很實用。

根據預設，VACUUM DELETE ONLY 回收空間，如此至少有 95% 的剩餘資料列不會標記為要進行刪除。若要變更單一資料表預設的刪除閾值，則在執行 VACUUM 時，包含資料表名稱和 TO threshold PERCENT 參數。

某些操作 (如 ALTER TABLE APPEND) 可能造成資料表遭切割成片段。當您使用 DELETE ONLY 子句時，清空操作會回收片段資料表的空間。重組操作同樣會套用 95% 的閾值。

REINDEX tablename

分析交錯排序索引鍵資料欄中值的分佈，然後執行完整的 VACUUM 操作。若使用了 REINDEX，則需有資料表名稱。

VACUUM REINDEX 所需的時間會比 VACUUM FULL 大幅增加，因為它會額外進行交錯排序索引鍵分析。交錯資料表的排序和合併操作可能需要更長的時間，因為交錯排序可能需要重新排列比複合排序更多的資料列。

如果 VACUUM REINDEX 操作在完成之前終止，下一次 VACUUM 在執行完整清空操作之前，會繼續執行重建索引操作。

TO threshold PERCENT 不支援 VACUUM REINDEX。

table_name

要清空的資料表名稱。如果您未指定資料表名稱，則清空操作會套用至目前資料庫中的所有資料表。您可以指定任何使用者建立的永久或臨時資料表。此命令對於其他物件並無意義，例如檢視和系統資料表。

若您包含 TO threshold PERCENT 參數，則需有資料表名稱。

RECLUSTER tablename

排序資料表中未排序的部分。已依自動資料表排序排序的資料表部分會保持不變。此命令不會將新排序的資料與已排序的區域合併。也不會回收標記為刪除的所有空間。完成此命令後，資料表可能不會顯示完全排序，如 SVV_TABLE_INFO 中的 unsorted 欄位所指示。

我們建議您針對頻繁擷取的大型資料表以及僅存取最新資料的查詢，使用 VACUUM RECLUSTER。

TO threshold PERCENT 不支援 VACUUM RECLUSTER。若使用了 RECLUSTER，則需有資料表名稱。

在具有交錯排序索引鍵的資料表和具有 ALL 分佈樣式的資料表上，不支援 VACUUM RECLUSTER。

table_name

要清空的資料表名稱。您可以指定任何使用者建立的永久或臨時資料表。此命令對於其他物件並無意義，例如檢視和系統資料表。

TO threshold PERCENT

此子句會指定閾值，超過此值時，VACUUM 會略過排序階段，以及指定在刪除階段中回收空間的目標閾值。排序閾值是清空之前，指定資料列中已依照排序順序排列的總列數百分比。 刪除閾值是清空之後，未標記為要進行刪除的總列數百分比下限。

由於 VACUUM 只會在資料表中已排序資料列的百分比低於排序閾值時重新排序資料列，因此 Amazon Redshift 通常可大幅縮短 VACUUM 的時間。同樣地，當 VACUUM 未限於回收全部標記為要刪除之資料列的空間時，它經常能夠略過重新寫入只包含少數已刪除資料列的區塊。

例如，若您指定 75 做為閾值，VACUUM 就會在資料表中有 75% 以上的資料列已依照排序順序排列時，略過排序階段。若是刪除階段，VACUUMS 會設定回收磁碟空間的目標，如此在清空之後，資料表中至少有 75% 的資料列不會標記為要進行刪除。閾值必須是介於 0 到 100 之間的整數。預設為 95。若您指定的值是 100，則除非資料表已完整排序，否則 VACUUM 一律會排序資料表，並且回收所有標記為要進行刪除之資料列的空間。若您指定的值是 0，則 VACUUM 永遠不會排序資料表，也不會回收空間。

若您包含 TO threshold PERCENT 參數，則也必須指定資料表名稱。若省略資料表名稱，則 VACUUM 會失敗。

您無法使用 TO threshold PERCENT 參數與 REINDEX 搭配。

BOOST

搭配其他資源 (例如記憶體和磁碟空間) 執行 VACUUM 命令 (可用時)。使用 BOOST 選項，VACUUM 會在一個視窗中運作，並在 VACUUM 操作的期間封鎖同時進行的刪除和更新。搭配 BOOST 選項執行會和系統資源發生競爭，可能影響查詢效能。請在系統負荷較輕時執行 VACUUM BOOST，例如維護操作期間。

請在使用 BOOST 選項時考慮以下事項：

- 指定 BOOST 時，table_name 值為必要項目。
- 不支援與 REINDEX 同時使用 BOOST。
- 使用 DELETE ONLY 時會忽略 BOOST。

使用須知

對於大多數 Amazon Redshift 應用程式，建議您執行完整清空。如需詳細資訊，請參閱 清空資料表。

在執行清空操作之前，請注意以下行為：

- 您無法執行 VACUUM 於交易區塊內 (BEGIN ... END)。如需交易的相關資訊，請參閱 可序列化隔離。
- 您任何時候都只能在叢集上執行一個 VACUUM 命令。若您嘗試同時執行多個清空操作，Amazon Redshift 會傳回錯誤。
- 當資料庫清空時，資料庫可能會稍微擴大。在沒有刪除的資料列可回收，或資料表的新排序順序導致資料壓縮比率降低時，這是預期的行為。
- 在清空操作期間，查詢效能預期會受到某種程度的影響。清空操作完成後，就會恢復正常效能。
- 在清空操作期間，同時的寫入操作會繼續執行，但不建議在清空時執行寫入操作。較有效率的方式，是在執行清空之前，先完成寫入操作。此外，在清空操作開始之後寫入的任何資料，都無法藉由該次操作清空。在此情況下，將需要執行第二次清空操作。
- 若載入或插入操作已在進行中，則清空操作可能無法開始執行。清空操作會暫時需要資料表的獨佔存取權，以開始執行。需要此獨佔存取權的時間很短，因此清空操作不會長時間封鎖同時執行的載入和插入操作。
- 若沒有可對特殊資料表執行的工作，則會略過清空操作；不過，在發現可略過該操作的過程中，可能會產生一些額外成本。若您知道這是原始資料表，或未達清空閾值，則不要對它執行清空操作。
- 在小型資料表上執行 DELETE ONLY 清空操作可能不會減少用來儲存資料的區塊數目，尤其是在資料表有大量資料欄，或叢集在每個節點上使用大量分割時。這些清空操作會對每個分割的每個資料欄新增一個區塊，用來負責同時插入資料表的工作，而此額外負荷有可能超過回收磁碟空間所減少的區塊數。例如，若 8 個節點叢集上的 10 欄資料表在清空前佔用了 1000 個區塊，則清空將不會減少實際的區塊數，除非因為刪除的資料列而回收了超過 80 個區塊的磁碟空間 (每個資料區塊都使用 1 MB)。

如果符合以下任何條件，則自動清空操作會暫停：

- 使用者執行資料定義語言 (DDL) 操作，例如 ALTER TABLE，其需要在自動清空目前作業所在的資料表上進行獨佔鎖定。
- 使用者在叢集中的任何資料表上觸發 VACUUM (一次只能執行一個 VACUUM)。
- 高叢集負載的期間。

範例

根據預設的 95% 清空閾值回收空間和資料庫，並排序所有資料表中的資料列。

```
vacuum;
```

回收空間，並根據預設的 95% 清空閾值重新排序 SALES 資料表中的資料列。

```
vacuum sales;
```

一律回收空間並重新排序 SALES 資料表中的資料列。

```
vacuum sales to 100 percent;
```

只有在已排序的資料列少於 75% 時，才重新排序 SALES 資料表中的資料列。

```
 vacuum sort only sales to 75 percent;
```

回收 SALES 資料表中的空間，如此至少有 75% 的剩餘資料列不會在清空後標記為要進行刪除。

```
vacuum delete only sales to 75 percent;
```

重新建立索引，然後清空 LISTING 資料表。

```
vacuum reindex listing;
```

下列命令會傳回錯誤。

```
vacuum reindex listing to 75 percent;
```

重新建立叢集，然後清空 LISTING 資料表。

```
vacuum recluster listing;
```

重新建立叢集，然後使用 BOOST 選項清空 LISTING 資料表。

```
vacuum recluster listing boost;
```

SQL 函數參考

主題

Amazon Redshift 支援許多 SQL 標準延伸的函數，以及標準彙總函數、純量函數和範圍函數。

> **Note**
>
> Amazon Redshift 是以 PostgreSQL 為基礎。在設計和開發您的資料倉儲應用程式時，您必須知道 Amazon Redshift 與 PostgreSQL 之間有多項重要的差異。如需 Amazon Redshift SQL 與 PostgreSQL 之間差異的相關資訊，請參閱 Amazon Redshift 和 PostgreSQL。

僅限領導節點函數

某些 Amazon Redshift 查詢會分配到運算節點上執行；其他的查詢則是只在領導者節點上執行。

當查詢參照使用者建立的資料表或系統資料表 (具有 STL 或 STV 字首的資料表，以及具有 SVL 或 SVV 字首的系統檢視) 時，領導者節點就會將 SQL 分送到運算節點。查詢如果只參考目錄資料表 (具有 PG 字首的資料表，例如 PG_TABLE_DEF)，或是未參考任何資料表，就只會在領導節點上執行。

某些 Amazon Redshift SQL 函數只有在領導者節點上才支援，在運算節點上不支援。使用領導者節點函數的查詢必須完全在領導者節點上執行，而不是在運算節點上，否則會傳回錯誤。

每個僅限於領導者節點的函數在文件中包含備註，說明如果函數參照使用者定義的資料表或 Amazon Redshift 系統資料表，將會傳回錯誤。

如需詳細資訊，請參閱 領導節點上所支援的 SQL 函數。

下列 SQL 函數是僅限於領導者節點的函數，在運算節點上不支援：

系統資訊函數

- CURRENT_SCHEMA
- CURRENT_SCHEMAS
- HAS_DATABASE_PRIVILEGE
- HAS_SCHEMA_PRIVILEGE
- HAS_TABLE_PRIVILEGE

字串函數

- SUBSTR

數學函數

- 階乘 (!)

下列僅限於領導者節點的函數已被棄用並且不再受支援：

日期函數

- AGE
- CURRENT_TIME
- CURRENT_TIMESTAMP
- LOCALTIME
- ISFINITE
- NOW

字串函數

- GETBIT
- GET_BYTE
- SET_BIT
- SET_BYTE
- TO_ASCII

僅限運算節點函數

某些 Amazon Redshift 查詢只能在運算節點上執行。如果查詢參考的是使用者建立的資料表，SQL 會在運算節點上執行。

查詢如果只參考目錄資料表 (具有 PG 字首的資料表，例如 PG_TABLE_DEF)，或是未參考任何資料表，就只會在領導節點上執行。

如果使用運算節點函數的查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表傳回下列錯誤。

```
[Amazon](500310) Invalid operation: One or more of the used functions must be applied
 on at least one user created table.
```

每個僅限於運算節點的函數在文件中包含備註，說明如果查詢未參照使用者定義的資料表或 Amazon Redshift 系統資料表，將會傳回錯誤。

下列 SQL 函數為僅限於運算節點的函數：

- LISTAGG

- MEDIAN
- PERCENTILE_CONT
- PERCENTILE_DISC 和 APPROXIMATE PERCENTILE_DISC

彙總函數

主題

- ANY_VALUE 函數
- APPROXIMATE PERCENTILE_DISC 函數
- AVG 函數
- COUNT 函數
- LISTAGG 函數
- MAX 函數
- MEDIAN 函數
- MIN 函數
- PERCENTILE_CONT 函數
- STDDEV_SAMP 和 STDDEV_POP 函數
- SUM 函數
- VAR_SAMP 和 VAR_POP 函數

彙總函數從一組輸入值計算單一結果值。

使用彙總函數的 SELECT 陳述式可以包含兩個選用子句：GROUP BY 和 HAVING。以下是這些子句的語法 (以 COUNT 函數為範例)：

```
SELECT count (*) expression FROM table_reference
WHERE condition [GROUP BY expression ] [ HAVING condition]
```

GROUP BY 子句依一或多個指定欄中的唯一值來彙總和分組結果。HAVING 子句將傳回的結果限定於特定彙總條件為 true 的列，例如 count (*) > 1。HAVING 子句的使用方式與 WHERE 根據欄的值來限定列一樣。如需這些額外子句的範例，請參閱 COUNT。

彙總函數不接受巢狀彙總函數或範圍函數做為引數。

ANY_VALUE 函數

ANY_VALUE 函數從輸入運算式值非確定性傳回任何值。如果輸入表達式不會傳回任何資料列，此函數會傳回 NULL。如果輸入表達式中有 NULL 值，函數也會傳回 NULL。

語法

```
ANY_VALUE( [ DISTINCT | ALL ] expression )
```

引數

DISTINCT | ALL

指定 DISTINCT 或 ALL 可從輸入運算式值傳回任何值。DISTINCT 引數沒有任何作用，而且會被忽略。

expression

函數運算的目標欄或運算式。expression 是下列其中一種資料類型：

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- DOUBLE PRECISION
- BOOLEAN
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

- VARBYTE
- SUPER
- HLLSKETCH
- GEOMETRY
- GEOGRAPHY

傳回值

傳回與 expression 相同的資料類型。

使用須知

如果指定欄 ANY_VALUE 函數的陳述式也包含第二個欄參考，則第二個欄必須出現在 GROUP BY 子句中，或包含在彙總函數中。

範例

這些範例使用《Amazon Redshift 入門指南》中的步驟 4：從 Amazon S3 載入範例資料中建立的事件表格。下列範例會傳回事件名稱為 Eagles 的任何 dateid 的執行個體。

```
select any_value(dateid) as dateid, eventname from event where eventname ='Eagles'
 group by eventname;
```

以下是結果。

```
dateid | eventname
-------+---------------
 1878  | Eagles
```

下列範例會傳回事件名稱為 Eagles 或 Cold War Kids 的任何 dateid 的執行個體。

```
select any_value(dateid) as dateid, eventname from event where eventname in('Eagles',
 'Cold War Kids') group by eventname;
```

以下是結果。

```
dateid | eventname
-------+---------------
```

```
 1922  | Cold War Kids
 1878  | Eagles
```

APPROXIMATE PERCENTILE_DISC 函數

APPROXIMATE PERCENTILE_DISC 是採用離散分佈模型的反向分佈函數。它採用百分位數值和排序規格，且會傳回給定集裡的一個元素。近似法可讓函數執行較快，其相對錯誤率低到約 0.5%。

對於給定的百分位數值，APPROXIMATE PERCENTILE_DISC 使用分位數摘要演算法，大致估計 ORDER BY 子句中的表達式的離散百分位數。APPROXIMATE PERCENTILE_DISC 傳回的值具有大於或等於百分位數的最小累積分佈值 (根據相同的排序規格)。

APPROXIMATE PERCENTILE_DISC 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。

語法

```
APPROXIMATE  PERCENTILE_DISC ( percentile )
WITHIN GROUP (ORDER BY expr)
```

引數

percentile

介於 0 和 1 之間的數值常數。計算時會忽略 Null。

WITHIN GROUP (ORDER BY expr)

此子句指定要排序和計算百分位數的數值或日期/時間值。

傳回值

資料類型與 WITHIN GROUP 子句中的 ORDER BY 表達式相同。

使用須知

如果 APPROXIMATE PERCENTILE_DISC 陳述式包含 GROUP BY 子句，則會限制結果集。限制會根據節點類型和節點數目而不同。如果超出限制，函數會失敗並傳回下列錯誤。

```
GROUP BY limit for approximate percentile_disc exceeded.
```

如果您需要評估的組數超過限制所允許，請考慮使用 PERCENTILE_CONT 函數。

範例

下列範例傳回排名前 10 個日期的銷售數量、銷售總計及第五個百分位數值。

```
select top 10 date.caldate,
count(totalprice), sum(totalprice),
approximate percentile_disc(0.5)
within group (order by totalprice)
from listing
join date on listing.dateid = date.dateid
group by date.caldate
order by 3 desc;

caldate    | count | sum        | percentile_disc
-----------+-------+------------+----------------
2008-01-07 |   658 | 2081400.00 |         2020.00
2008-01-02 |   614 | 2064840.00 |         2178.00
2008-07-22 |   593 | 1994256.00 |         2214.00
2008-01-26 |   595 | 1993188.00 |         2272.00
2008-02-24 |   655 | 1975345.00 |         2070.00
2008-02-04 |   616 | 1972491.00 |         1995.00
2008-02-14 |   628 | 1971759.00 |         2184.00
2008-09-01 |   600 | 1944976.00 |         2100.00
2008-07-29 |   597 | 1944488.00 |         2106.00
2008-07-23 |   592 | 1943265.00 |         1974.00
```

AVG 函數

AVG 函數傳回輸入表達式值的平均值 (算術平均數)。AVG 函數處理數值，且忽略 NULL 值。

語法

```
AVG ( [ DISTINCT | ALL ] expression )
```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT

- INTEGER
- BIGINT
- NUMERIC
- DECIMAL
- REAL
- DOUBLE PRECISION
- SUPER

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算平均值之前，將從指定的表達式中消除所有重複值。如果指定引數 ALL，則函數在計算平均值時會保留表達式中的所有重複值。ALL 為預設值。

資料類型

AVG 函數支援的引數類型包括
SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL、DOUBLE PRECISION 及 SUPER。

AVG 函數支援的傳回類型如下：

- BIGINT 代表任何整數類型引數
- DOUBLE PRECISION 代表浮點數引數
- 針對任何其他引數類型，傳回與運算式相同的資料類型。

具有 NUMERIC 或 DECIMAL 引數的 AVG 函數，結果的預設精確度為 38。結果的小數位數和引數的小數位數相同。例如，DEC(5,2) 欄的 AVG 會傳回 DEC(38,2) 資料類型。

範例

從 SALES 資料表尋找每次交易的平均銷售數量：

```
select avg(qtysold)from sales;

avg
-----
2
(1 row)
```

尋找所有列表所列出的平均總價：

```
select avg(numtickets*priceperticket) as avg_total_price from listing;

avg_total_price
-----------------
3034.41
(1 row)
```

尋找平均支付價格，按月份分組，依遞減順序排列：

```
select avg(pricepaid) as avg_price, month
from sales, date
where sales.dateid = date.dateid
group by month
order by avg_price desc;

avg_price | month
-----------+-------
659.34 | MAR
655.06 | APR
645.82 | JAN
643.10 | MAY
642.72 | JUN
642.37 | SEP
640.72 | OCT
640.57 | DEC
635.34 | JUL
635.24 | FEB
634.24 | NOV
632.78 | AUG
(12 rows)
```

COUNT 函數

COUNT 函數計算表達式所定義的列數。

COUNT 函數有下列版本。

- COUNT (*) 計算目標資料表中的所有列數，而不論是否包含 Null。
- COUNT (expression) 計算特定欄或表達式中不含 NULL 值的列數。
- COUNT (DISTINCT expression) 計算某欄或表達式中相異非 NULL 值的個數。

- APPROXIMATE COUNT DISTINCT 會大致估計某欄或運算式中相異非 NULL 值的個數。

語法

```
COUNT( * | expression )
```

```
COUNT ( [ DISTINCT | ALL ] expression )
```

```
APPROXIMATE COUNT ( DISTINCT expression )
```

引數

expression

函數運算的目標欄或表達式。COUNT 函數支援所有引數資料類型。

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計數之前會從指定的表達式中消除所有重複值。如果指定引數 ALL，則函數在計數時會保留表達式中的所有重複值。ALL 為預設值。

APPROXIMATE

當與近似使用時，COUNT DISTINCT 函數會使用 HyperLogLog 演算法來近似資料行或運算式中不同非空值的數目。使用 APPROXIMATE 關鍵字的查詢執行較快，其相對錯誤率低到約 2%。如果每個查詢或每一組 (若有 group by 子句) 傳回數百萬個以上的大量相異值，則查詢一定要採用近似法。如果相異值較少 (數千個)，則近似法可能比精確計數更慢。APPROXIMATE 只能與 COUNT DISTINCT 一起使用。

傳回類型

COUNT 函數傳回 BIGINT。

範例

計算佛羅里達州的所有使用者人數：

```
select count(*) from users where state='FL';
```

```
count
-------
510
```

計算 EVENT 表中的所有事件名稱：

```
select count(eventname) from event;

count
-------
8798
```

計算 EVENT 表中的所有事件名稱：

```
select count(all eventname) from event;

count
-------
8798
```

從 EVENT 資料表計算所有唯一會場 ID 的數目：

```
select count(distinct venueid) as venues from event;

venues
--------
204
```

計算每個賣方列出整批銷售門票超過四張的次數。結果依賣方 ID 分組：

```
select count(*), sellerid from listing
where numtickets > 4
group by sellerid
order by 1 desc, 2;

count | sellerid
------+----------
12    |    6386
11    |    17304
11    |    20123
11    |    25428
```

```
...
```

下列範例比較 COUNT 和 APPROXIMATE COUNT 的傳回值和執行時間。

```
select  count(distinct pricepaid) from sales;

count
-------
  4528


Time: 48.048 ms


select approximate count(distinct pricepaid) from sales;

count
-------
  4553


Time: 21.728 ms
```

LISTAGG 函數

對於查詢中的每一組，LISTAGG 彙整函數依據 ORDER BY 表達式來排序該組的列，然後將這些值串連成單一字串。

LISTAGG 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。如需詳細資訊，請參閱 查詢目錄資料表。

語法

```
LISTAGG( [DISTINCT] aggregate_expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]
```

引數

DISTINCT

此子句在串連值之前會從指定的運算式中消除重複值。忽略結尾空格。例如，字串 'a' 和 'a ' 被視為重複。LISTAGG 會使用第一個遇到的值。如需詳細資訊，請參閱 多餘空格的意義。

aggregate_expression

任何有效運算式 (例如欄名)，用於提供要彙總的值。忽略 NULL 值和空字串。

delimiter

用來區隔串連值的字串常數。預設值為 NULL。

WITHIN GROUP (ORDER BY order_list)

此子句指定彙總值的排序順序。

傳回值

VARCHAR(MAX)。如果結果集大於 VARCHAR 大小上限，則 LISTAGG 會傳回下列錯誤：

```
Invalid operation: Result size exceeds LISTAGG limit
```

使用須知

- 如果陳述式包含多個使用 WITHIN GROUP 子句的 LISTAGG 函數，則每一個 WITHIN GROUP 子句必須使用相同的 ORDER BY 值。

  例如，下列陳述式會傳回錯誤。

  ```
  SELECT LISTAGG(sellerid)
  WITHIN GROUP (ORDER BY dateid) AS sellers,
  LISTAGG(dateid)
  WITHIN GROUP (ORDER BY sellerid) AS dates
  FROM sales;
  ```

  下列陳述式會成功執行。

  ```
  SELECT LISTAGG(sellerid)
  WITHIN GROUP (ORDER BY dateid) AS sellers,
  LISTAGG(dateid)
  WITHIN GROUP (ORDER BY dateid) AS dates
  FROM sales;

  SELECT LISTAGG(sellerid)
  WITHIN GROUP (ORDER BY dateid) AS sellers,
  LISTAGG(dateid) AS dates
  ```

```
FROM sales;
```

範例

下列範例彙總賣方 ID，依賣方 ID 排序。

```
SELECT LISTAGG(sellerid, ', ')
WITHIN GROUP (ORDER BY sellerid)
FROM sales
WHERE eventid = 4337;

listagg

-------------------------------------------------------------------------------------------------
380, 380, 1178, 1178, 1178, 2731, 8117, 12905, 32043, 32043, 32043, 32432, 32432,
 38669, 38750, 41498, 45676, 46324, 47188, 47188, 48294
```

下列範例使用 DISTINCT 傳回唯一賣方 ID 的清單。

```
SELECT LISTAGG(DISTINCT sellerid, ', ')
WITHIN GROUP (ORDER BY sellerid)
FROM sales
WHERE eventid = 4337;

listagg

-----------------------------------------------------------------------------------------
380, 1178, 2731, 8117, 12905, 32043, 32432, 38669, 38750, 41498, 45676, 46324, 47188,
 48294
```

下列範例彙總賣方 ID，依日期順序排序。

```
SELECT LISTAGG(sellerid, ', ')
WITHIN GROUP (ORDER BY dateid)
FROM sales
WHERE eventid = 4337;

   listagg
-------------------------------------------------------------------------------------------------
 41498, 47188, 47188, 1178, 1178, 1178, 380, 45676, 46324, 48294, 32043, 32043, 32432,
 12905, 8117, 38750, 2731, 32432, 32043, 380, 38669
```

下列範例以縱線分隔清單傳回 ID 為 660 的買方的銷售日期。

```
SELECT LISTAGG(
    (SELECT caldate FROM date WHERE date.dateid=sales.dateid), ' | '
)
WITHIN GROUP (ORDER BY sellerid DESC, salesid ASC)
FROM sales
WHERE buyerid = 660;

             listagg
-------------------------------------------------
2008-07-16 | 2008-07-09 | 2008-01-01 | 2008-10-26
```

下列範例以逗號分隔清單傳回買方 ID 660、661 和 662 的銷售 ID。

```
SELECT buyerid,
LISTAGG(salesid,', ')
WITHIN GROUP (ORDER BY salesid) AS sales_id
FROM sales
WHERE buyerid BETWEEN 660 AND 662
GROUP BY buyerid
ORDER BY buyerid;

buyerid |                sales_id
--------+-------------------------------------------------------
660     | 32872, 33095, 33514, 34548
661     | 19951, 20517, 21695, 21931
662     | 3318, 3823, 4215, 51980, 53202, 55908, 57832, 171603
```

MAX 函數

MAX 函數傳回一個列集的最大值。可使用 DISTINCT 或 ALL，但不影響結果。

語法

```
MAX ( [ DISTINCT | ALL ] expression )
```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- DOUBLE PRECISION
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- VARBYTE
- SUPER

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算最大值之前，將從指定的表達式中消除所有重複值。如果指定引數 ALL，則函數在計算最大值時會保留表達式中的所有重複值。ALL 為預設值。

資料類型

傳回與 expression 相同的資料類型。MIN 函數的布林同等函數為 BOOL_AND 函數，MAX 的布林同等函數為 BOOL_OR 函數。

範例

從所有銷售中尋找最高支付價格：

```
select max(pricepaid) from sales;

max
----------
12624.00
(1 row)
```

從所有銷售中尋找每張門票的最高支付價格：

```
select max(pricepaid/qtysold) as max_ticket_price
from sales;

max_ticket_price
-----------------
2500.00000000
(1 row)
```

MEDIAN 函數

計算值範圍的中值。忽略範圍內的 NULL 值。

MEDIAN 是採用連續分佈模型的反向分佈函數。

MEDIAN 是 PERCENTILE_CONT 的特殊情況。

MEDIAN 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。

語法

```
MEDIAN(median_expression)
```

引數

median_expression

函數運算的目標欄或表達式。

資料類型

傳回類型取決於 median_expression 的資料類型。下表顯示每一個 median_expression 資料類型的傳回類型。

輸入類型	傳回類型
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL
FLOAT, DOUBLE	DOUBLE

輸入類型	傳回類型
DATE	DATE
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMPTZ

使用須知

如果 median_expression 引數是以最大精確度 38 位數定義的 DECIMAL 資料類型，MEDIAN 可能會傳回不準確的結果或錯誤。如果 MEDIAN 函數的傳回值超過 38 位數，會將結果截斷為適合長度，導致精確度降低。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些情況，建議使用精確度較低的資料類型，或將 median_expression 引數轉換為較低精確度。

如果陳述式中多次呼叫會排序的彙總函數 (LISTAGG、PERCENTILE_CONT 或 MEDIAN)，則所有呼叫必須使用相同的 ORDER BY 值。請注意，MEDIAN 會對表達式值套用隱含的 order by。

例如，下列陳述式會傳回錯誤。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;

An error occurred when executing the SQL command:
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;

ERROR: within group ORDER BY clauses for aggregate functions must be the same
```

下列陳述式會成功執行。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
```

```
MEDIAN(salesid)
FROM sales
GROUP BY salesid, pricepaid;
```

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

下列範例顯示 MEDIAN 產生與 PERCENTILE_CONT(0.5) 相同的結果。

```
SELECT TOP 10 DISTINCT sellerid, qtysold,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold),
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid, qtysold;

+----------+---------+-----------------+--------+
| sellerid | qtysold | percentile_cont | median |
+----------+---------+-----------------+--------+
|        2 |       2 |               2 |      2 |
|       26 |       1 |               1 |      1 |
|       33 |       1 |               1 |      1 |
|       38 |       1 |               1 |      1 |
|       43 |       1 |               1 |      1 |
|       48 |       2 |               2 |      2 |
|       48 |       3 |               3 |      3 |
|       77 |       4 |               4 |      4 |
|       85 |       4 |               4 |      4 |
|       95 |       2 |               2 |      2 |
+----------+---------+-----------------+--------+
```

下列範例會找出每個 sellerid 售出的中位數量。

```
SELECT sellerid,
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid
ORDER BY sellerid
LIMIT 10;

+----------+--------+
| sellerid | median |
+----------+--------+
```

```
|        1 |    1.5 |
|        2 |      2 |
|        3 |      2 |
|        4 |      2 |
|        5 |      1 |
|        6 |      1 |
|        7 |    1.5 |
|        8 |      1 |
|        9 |      4 |
|       12 |      2 |
+----------+--------+
```

若要驗證第一個 sellerid 的先前查詢結果，請使用下列範例。

```
SELECT qtysold
FROM sales
WHERE sellerid=1;

+---------+
| qtysold |
+---------+
|       2 |
|       1 |
+---------+
```

MIN 函數

MIN 函數傳回一個列集的最小值。可使用 DISTINCT 或 ALL，但不影響結果。

語法

```
MIN ( [ DISTINCT | ALL ] expression )
```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT
- INTEGER
- BIGINT

- DECIMAL
- REAL
- DOUBLE PRECISION
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- VARBYTE
- SUPER

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算最小值之前，將從指定的表達式中消除所有重複值。如果指定引數 ALL，則函數在計算最小值時會保留表達式中的所有重複值。ALL 為預設值。

資料類型

傳回與 expression 相同的資料類型。MIN 函數的布林同等函數為 BOOL_AND 函數，MAX 的布林同等函數為 BOOL_OR 函數。

範例

從所有銷售中尋找最低支付價格：

```
select min(pricepaid) from sales;

min
-------
20.00
(1 row)
```

從所有銷售中尋找每張門票的最低支付價格：

```
select min(pricepaid/qtysold)as min_ticket_price
from sales;
```

```
min_ticket_price
------------------
20.00000000
(1 row)
```

PERCENTILE_CONT 函數

PERCENTILE_CONT 是採用連續分佈模型的反向分佈函數。它採用百分位數值和排序規格，且會傳回插入值，該值將根據排序規格落入給定的百分位數值。

PERCENTILE_CONT 在值排序後計算值之間的線性插值。此函數在列根據排序規格來排序後，使用彙總群組中的百分位數值 (P) 和非 Null 列數 (N) 來計算列號。此列號 (RN) 是根據公式 RN = (1+ (P*(N-1)) 來計算。彙總函數的最終結果是以列號 CRN = CEILING(RN) 到 FRN = FLOOR(RN) 各列的值之間的線性插值來計算。

最終結果如下。

如果 (CRN = FRN = RN)，則結果為 (value of expression from row at RN)

否則結果如下：

(CRN - RN) * (value of expression for row at FRN) + (RN - FRN) * (value of expression for row at CRN).

PERCENTILE_CONT 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。

語法

```
PERCENTILE_CONT(percentile)
WITHIN GROUP(ORDER BY expr)
```

引數

percentile

介於 0 到 1 之間的數值常數。計算中會忽略 NULL 值。

expr

指定要排序和計算百分位數的數值或日期/時間值。

傳回值

傳回類型取決於 WITHIN GROUP 子句中 ORDER BY 表達式的資料類型。下表顯示每一個 ORDER BY 表達式資料類型的傳回類型。

輸入類型	傳回類型
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL
FLOAT, DOUBLE	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMPTZ

使用須知

如果 ORDER BY 表達式是以最大精確度 38 位數定義的 DECIMAL 資料類型，PERCENTILE_CONT 可能會傳回不準確的結果或錯誤。如果 PERCENTILE_CONT 函數的傳回值超過 38 位數，結果會截斷為適合長度，導致精確度降低。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些情況，建議使用精確度較低的資料類型，或將 ORDER BY 表達式轉換為較低精確度。

如果陳述式中多次呼叫會排序的彙總函數 (LISTAGG、PERCENTILE_CONT 或 MEDIAN)，則所有呼叫必須使用相同的 ORDER BY 值。請注意，MEDIAN 會對表達式值套用隱含的 order by。

例如，下列陳述式會傳回錯誤。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;

An error occurred when executing the SQL command:
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
```

```
FROM sales
GROUP BY salesid, pricepaid;

ERROR: within group ORDER BY clauses for aggregate functions must be the same
```

下列陳述式會成功執行。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(salesid)
FROM sales
GROUP BY salesid, pricepaid;
```

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

下列範例顯示 PERCENTILE_CONT(0.5) 產生與 MEDIAN 相同的結果。

```
SELECT TOP 10 DISTINCT sellerid, qtysold,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold),
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid, qtysold;

+----------+---------+-----------------+--------+
| sellerid | qtysold | percentile_cont | median |
+----------+---------+-----------------+--------+
|        2 |       2 |               2 |      2 |
|       26 |       1 |               1 |      1 |
|       33 |       1 |               1 |      1 |
|       38 |       1 |               1 |      1 |
|       43 |       1 |               1 |      1 |
|       48 |       2 |               2 |      2 |
|       48 |       3 |               3 |      3 |
|       77 |       4 |               4 |      4 |
|       85 |       4 |               4 |      4 |
|       95 |       2 |               2 |      2 |
+----------+---------+-----------------+--------+
```

以下範例會找出 SALES 表格中每個 sellerid 的銷售數量的 PERCENTILE_CONT(0.5) 和 PERCENTILE_CONT(0.75)。

```
SELECT sellerid,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold) as pct_05,
PERCENTILE_CONT(0.75) WITHIN GROUP(ORDER BY qtysold) as pct_075
FROM sales
GROUP BY sellerid
ORDER BY sellerid
LIMIT 10;

+----------+--------+---------+
| sellerid | pct_05 | pct_075 |
+----------+--------+---------+
|        1 |    1.5 |    1.75 |
|        2 |      2 |    2.25 |
|        3 |      2 |       3 |
|        4 |      2 |       2 |
|        5 |      1 |     1.5 |
|        6 |      1 |       1 |
|        7 |    1.5 |    1.75 |
|        8 |      1 |       1 |
|        9 |      4 |       4 |
|       12 |      2 |    3.25 |
+----------+--------+---------+
```

若要驗證第一個 sellerid 的先前查詢結果，請使用下列範例。

```
SELECT qtysold
FROM sales
WHERE sellerid=1;

+---------+
| qtysold |
+---------+
|       2 |
|       1 |
+---------+
```

STDDEV_SAMP 和 STDDEV_POP 函數

STDDEV_SAMP 和 STDDEV_POP 函數傳回一組數值 (整數、小數或浮點數) 的樣本標準差和母體標準差。STDDEV_SAMP 函數的結果相當於同一組值的樣本變異數平方根。

STDDEV_SAMP 和 STDDEV 是同一個函數的同義詞。

語法

```
STDDEV_SAMP | STDDEV ( [ DISTINCT | ALL ] expression)
STDDEV_POP ( [ DISTINCT | ALL ] expression)
```

表達式必須為整數、小數或浮點數資料類型。不論表達式的資料類型為何，此函數的傳回類型都是雙精確度數字。

> **Note**
>
> 標準差是採用浮點運算來計算，所得結果可能稍不精確。

使用須知

對包含單一值的表達式計算樣本標準差 (STDDEV 或 STDDEV_SAMP) 時，函數的結果為 NULL，不是 0。

範例

下列查詢傳回 VENUE 資料表的 VENUESEATS 欄中各值的平均值，接著傳回同一組值的樣本標準差和母體標準差。VENUESEATS 是 INTEGER 欄。結果的小數位數簡化到 2 位數。

```
select avg(venueseats),
cast(stddev_samp(venueseats) as dec(14,2)) stddevsamp,
cast(stddev_pop(venueseats) as dec(14,2)) stddevpop
from venue;

avg  | stddevsamp | stddevpop
-------+------------+-----------
17503 |   27847.76 |  27773.20
(1 row)
```

下列查詢傳回 SALES 資料表中的 COMMISSION 欄的樣本標準差。COMMISSION 是 DECIMAL 欄。結果的小數位數簡化到 10 位數。

```
select cast(stddev(commission) as dec(18,10))
from sales;

stddev
```

```
----------------
130.3912659086
(1 row)
```

下列查詢將 COMMISSION 欄的樣本標準差轉換為整數。

```
select cast(stddev(commission) as integer)
from sales;

stddev
--------
130
(1 row)
```

下列查詢傳回 COMMISSION 欄的樣本標準差和樣本變異數平方根。這些計算的結果相同。

```
select
cast(stddev_samp(commission) as dec(18,10)) stddevsamp,
cast(sqrt(var_samp(commission)) as dec(18,10)) sqrtvarsamp
from sales;

stddevsamp   |  sqrtvarsamp
----------------+----------------
130.3912659086 | 130.3912659086
(1 row)
```

SUM 函數

SUM 函數傳回輸入欄或表達式值的總和。SUM 函數處理數值，且忽略 NULL 值。

語法

```
SUM ( [ DISTINCT | ALL ] expression )
```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT

- INTEGER
- BIGINT
- NUMERIC
- DECIMAL
- REAL
- DOUBLE PRECISION
- SUPER

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算總和之前，將從指定的表達式中消除所有重複值。如果指定引數 ALL，則函數在計算總和時會保留表達式中的所有重複值。ALL 為預設值。

資料類型

SUM 函數支援的引數類型包括
SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL、DOUBLE PRECISION 及 SUPER。

SUM 函數支援的傳回類型如下

- BIGINT 代表 BIGINT、SMALLINT 及 INTEGER 引數
- NUMERIC 代表 NUMERIC 引數
- DOUBLE PRECISION 代表浮點數引數
- 針對任何其他引數類型，傳回與運算式相同的資料類型。

具有 NUMERIC 或 DECIMAL 引數的 SUM 函數，結果的預設精確度為 38。結果的小數位數和引數的小數位數相同。例如，DEC(5,2) 欄的 SUM 會傳回 DEC(38,2) 資料類型。

範例

從 SALES 資料表中尋找所有已付佣金的總和：

```
select sum(commission) from sales;

sum
-------------
16614814.65
```

```
(1 row)
```

尋找佛羅里達州的所有會場的座位數：

```
select sum(venueseats) from venue
where venuestate = 'FL';

sum
--------
250411
(1 row)
```

尋找五月售出的座位數：

```
select sum(qtysold) from sales, date
where sales.dateid = date.dateid and date.month = 'MAY';

sum
-------
32291
(1 row)
```

VAR_SAMP 和 VAR_POP 函數

VAR_SAMP 和 VAR_POP 函數傳回一組數值 (整數、小數或浮點數) 的樣本變異數和母體變異數。VAR_SAMP 函數的結果相當於同一組值的平方樣本標準差。

VAR_SAMP 和 VARIANCE 是同一個函數的同義詞。

語法

```
VAR_SAMP | VARIANCE ( [ DISTINCT | ALL ] expression)
VAR_POP ( [ DISTINCT | ALL ] expression)
```

表達式必須為整數、小數或浮點數資料類型。不論表達式的資料類型為何，此函數的傳回類型都是雙精確度數字。

> **Note**
>
> 這些函數的結果可能隨著資料倉儲叢集而有所不同，視每個案例中的叢集組態而定。

使用須知

對包含單一值的表達式計算樣本變異數 (VARIANCE 或 VAR_SAMP) 時，函數的結果為 NULL，不是 0。

範例

下列查詢傳回 LISTING 資料表中的 NUMTICKETS 欄的四捨五入樣本變異數和母體變異數。

```
select avg(numtickets),
round(var_samp(numtickets)) varsamp,
round(var_pop(numtickets)) varpop
from listing;

avg | varsamp | varpop
-----+---------+--------
10 |      54 |     54
(1 row)
```

下列查詢執行同樣的計算，但將結果轉換為小數值。

```
select avg(numtickets),
cast(var_samp(numtickets) as dec(10,4)) varsamp,
cast(var_pop(numtickets) as dec(10,4)) varpop
from listing;

avg | varsamp | varpop
-----+---------+---------
10 | 53.6291 | 53.6288
(1 row)
```

陣列函數

接下來，您可以找到 Amazon Redshift 支援存取和操作陣列的 SQL 陣列函數的說明。

主題

- 陣列函數
- array_concat 函數
- array_flatten 陣列

- get_array_length 陣列
- split_to_array 陣列
- 子陣列函數

陣列函數

建立 SUPER 資料類型的陣列。

語法

```
ARRAY( [ expr1 ] [ , expr2 [ , ... ] ] )
```

引數

expr1, expr2

任何 Amazon Redshift 資料類型的運算式 (日期和時間類型除外)，因為 Amazon Redshift 不會將日期和時間類型轉換為 SUPER 資料類型。引數不需要是相同的資料類型。

傳回類型

陣列函數傳回 SUPER 資料類型。

範例

下列範例顯示數值陣列和不同資料類型的陣列。

```
--an array of numeric values
select array(1,50,null,100);
      array
------------------
 [1,50,null,100]
(1 row)

--an array of different data types
select array(1,'abc',true,3.14);
        array
-----------------------
 [1,"abc",true,3.14]
```

```
(1 row)
```

array_concat 函數

array_concat 函數串連兩個陣列以建立一個陣列，其中包含第一個陣列中的所有元素，後跟第二個陣列中的所有元素。這兩個引數必須是有效的陣列。

語法

```
array_concat( super_expr1,  super_expr2 )
```

引數

super_expr1

指定要串連的兩個陣列中第一個的值。

super_expr2

指定要串連的兩個陣列中第二個的值。

傳回類型

array_concat 函數傳回 SUPER 資料值。

範例

下列範例顯示串連相同類型的兩個陣列，以及串連不同類型的兩個陣列。

```
-- concatenating two arrays
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY(10003,10004));
              array_concat
-----------------------------------
 [10001,10002,10003,10004]
(1 row)

-- concatenating two arrays of different types
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY('ab','cd'));
          array_concat
-----------------------------
 [10001,10002,"ab","cd"]
(1 row)
```

array_flatten 陣列

將多個陣列合併成 SUPER 類型的單個陣列。

語法

```
array_flatten( super_expr1,super_expr2,.. )
```

引數

super_expr1、super_expr2

陣列形式的有效 SUPER 運算式。

傳回類型

array_flatten 函數會傳回 SUPER 資料值。

範例

下列範例顯示 array_flatten 函數。

```
SELECT ARRAY_FLATTEN(ARRAY(ARRAY(1,2,3,4),ARRAY(5,6,7,8),ARRAY(9,10)));
     array_flatten
-----------------------
 [1,2,3,4,5,6,7,8,9,10]
(1 row)
```

get_array_length 陣列

傳回指定陣列的長度。GET_ARRAY_LENGTH 函數傳回給定的物件或陣列路徑的 SUPER 陣列的長度。

語法

```
get_array_length( super_expr )
```

引數

super_expr

陣列形式的有效 SUPER 運算式。

傳回類型

get_array_length 函數會傳回 BIGINT。

範例

下列範例顯示 get_array_length 函數。

```
SELECT GET_ARRAY_LENGTH(ARRAY(1,2,3,4,5,6,7,8,9,10));
 get_array_length
----------------------
            10
(1 row)
```

split_to_array 陣列

使用分隔符號做為選擇性的參數。如果沒有分隔符號，則預設值為逗號。

語法

```
split_to_array( string,delimiter )
```

引數

string

要分割的輸入字串。

delimiter

將根據其分割輸入字串的選用值。預設為逗號。

傳回類型

split_to_array 函數會傳回 SUPER 資料值。

範例

下列範例顯示 split_to_array 函數。

```
SELECT SPLIT_TO_ARRAY('12|345|6789', '|');
     split_to_array
------------------------
```

```
 ["12","345","6789"]
(1 row)
```

子陣列函數

操控陣列以傳回輸入陣列的子集。

語法

```
SUBARRAY( super_expr, start_position, length )
```

引數

super_expr

陣列形式的有效 SUPER 運算式。

start_position

陣列中開始擷取的位置，從索引位置 0 開始。負位置從陣列的末尾向後計數。

長度

要擷取的元素數 (子字串的長度)。

傳回類型

子陣列函數傳回 SUPER 資料值。

範例

下列為子陣列函數的範例。

```
 SELECT SUBARRAY(ARRAY('a', 'b', 'c', 'd', 'e', 'f'), 2, 3);
   subarray
---------------
 ["c","d","e"]
(1 row)
```

位元彙整函數

位元彙總函數會運算位元操作，以執行整數欄以及可轉換或四捨五入為整數值的彙總。

主題

在位元彙整中使用 NULL

對可為 Null 的欄套用位元函數時，計算函數結果之前會消除任何 NULL 值。如果沒有任何列符合彙總的資格，位元函數會傳回 Null。同樣行為適用於一般彙總函數。以下是範例。

```
select sum(venueseats), bit_and(venueseats) from venue
where venueseats is null;

sum  | bit_and
------+---------
null |    null
(1 row)
```

位元彙整的 DISTINCT 支援

如同其他彙總函數，位元函數也支援 DISTINCT 關鍵字。

不過，DISTINCT 與這些函數一起使用不影響結果。值的第一個實例足以滿足位元 AND 或 OR 運算。如果正在評估的運算式中存在重複值，則沒有任何區別。

因為 DISTINCT 處理很可能引起某些查詢執行負荷，因此我們建議您不要將 DISTINCT 與位元函數一起使用。

位元函數的概述範例

在下文中，您可以找到一些概述範例，展示如何使用位元函數。您還可以找到具有每個函數描述的特定代碼範例。

位元函數的範例是以 TICKIT 範例資料庫為基礎。TICKIT 範例資料庫中的 USERS 資料表包含幾個布林值欄，指出是否已知每一個使用者喜歡的各種類型活動，例如運動、戲劇、歌劇等。範例如下。

```
select userid, username, lastname, city, state,
likesports, liketheatre
from users limit 10;

userid | username | lastname  |     city     | state | likesports | liketheatre
-------+----------+-----------+--------------+-------+------------+-------------
1 | JSG99FHE | Taylor    | Kent         | WA    | t          | t
9 | MSD36KVR | Watkins   | Port Orford  | MD    | t          | f
```

假設 USERS 表格的新版本是以不同的方式建置的。在此新版本中，單一整數欄定義 (以二進位格式) 每一個使用者喜歡或不喜歡的八種活動。在這種設計中，每個位置代表一種類型的事件。喜歡所有八種類型的使用者將全部八位設定為 1 (如下表的第一列)。如果使用者不喜歡任何活動，則八個位元全部設為 0 (請看第二列)。接下來的第三列表示只喜歡運動和爵士樂的使用者。

	運動	戲劇	爵士樂	歌劇	搖滾	拉斯維加斯	百老匯	古典
使用者 1	1	1	1	1	1	1	1	1
使用者 2	0	0	0	0	0	0	0	0
使用者 3	1	0	1	0	0	0	0	0

在資料庫資料表中，這些二進位值可以作為整數儲存在單一 LIKES 欄中，如下所示。

使用者	二進位值	儲存的值 (整數)
使用者 1	11111111	255
使用者 2	00000000	0
使用者 3	10100000	160

BIT_AND 函數

BIT_AND 函數會對單一整數欄或表達式中的所有值執行位元 AND 操作。此函數會彙總每一個二進位值 (對應於表達式中的每一個整數值) 的每一個位元。

如果所有值之中沒有任何位元設為 1，BIT_AND 函數會傳回結果 0。如果所有值之中有一或多個位元設為 1，此函數會傳回整數值。此整數是對應於那些位元的二進位值的數字。

例如，資料表有一欄包含四個整數值：3、7、10 及 22。這些整數以二進位格式表示如下：

Integer	二進位值
3	11
7	111
10	1010
22	10110

此資料集上的 BIT_AND 作業會發現所有位元都設定為只1在 second-to-last 位置。結果為 00000010 的二進位值，代表整數值 2。因此，BIT_AND 函數會傳回 2。

語法

```
BIT_AND ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須為 INT、INT2 或 INT8 資料類型。此函數會傳回同等的 INT、INT2 或 INT8 資料類型。

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算結果之前，將消除指定之表達式的所有重複值。如果指定引數 ALL，則函數會保留所有重複值。ALL 為預設值。如需詳細資訊，請參閱 位元彙整的 DISTINCT 支援。

範例

由於整數欄中儲存有意義的商業資訊，您可以使用位元函數來擷取和彙總該資訊。下列查詢將 BIT_AND 函數套用至名為 USERLIKES 之資料表中的 LIKES 欄，並依 CITY 欄將結果分組。

```
select city, bit_and(likes) from userlikes group by city
order by city;
city          | bit_and
--------------+---------
Los Angeles   |       0
Sacramento    |       0
San Francisco |       0
San Jose      |      64
Santa Barbara |     192
(5 rows)
```

您可將這些結果解釋如下：

- Santa Barbara 的整數值 192 轉換為二進位值 11000000。換言之，此城市的所有使用者都喜歡運動和戲劇，但並非所有使用者還喜歡其他任何一種活動。
- 整數 64 轉換為 01000000。因此，對於 San Jose 中的使用者而言，他們全部都喜歡的一種活動只有戲劇。
- 其他三個城市的值為 0，表示那些城市的所有使用者沒有共同的「愛好」。

BIT_OR 函數

BIT_OR 函數會對單一整數欄或表達式中的所有值執行位元 OR 操作。此函數會彙總每一個二進位值 (對應於表達式中的每一個整數值) 的每一個位元。

例如，假設資料表有一欄包含四個整數值：3、7、10 及 22。這些整數以二進位格式表示如下：

Integer	二進位值
3	11
7	111
10	1010
22	10110

如果您將 BIT_OR 函數套用至整數值集，則作業會尋找在每個位置中找到 1 的任何值。在此案例中，至少一個值的最後五個位置有 1，於是產生二進位結果 00011111；因此，函數傳回 31 (亦即 16 + 8 + 4 + 2 + 1)。

語法

```
BIT_OR ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須為 INT、INT2 或 INT8 資料類型。此函數會傳回同等的 INT、INT2 或 INT8 資料類型。

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算結果之前，將消除指定之表達式的所有重複值。如果指定引數 ALL，則函數會保留所有重複值。ALL 為預設值。如需詳細資訊，請參閱 位元彙整的 DISTINCT 支援。

範例

下列查詢將 BIT_OR 函數套用至名為 USERLIKES 之資料表中的 LIKES 欄，並依 CITY 欄將結果分組。

```
select city, bit_or(likes) from userlikes group by city
order by city;
city          | bit_or
--------------+--------
Los Angeles   |    127
Sacramento    |    255
San Francisco |    255
San Jose      |    255
Santa Barbara |    255
(5 rows)
```

對於列出的其中四個城市，至少有一個使用者喜歡所有活動類型 (255=11111111)。以 Los Angeles 來說，至少有一個使用者喜歡所有活動類型，但運動除外 (127=01111111)。

BOOL_AND 函數

BOOL_AND 函數會對單一布林值或整數欄或表達式執行操作。此函數會將類似邏輯套用至 BIT_AND 和 BIT_OR 函數。此函數的傳回類型為布林值 (`true` 或 `false`)。

如果一組值全部為 true，BOOL_AND 函數會傳回 `true` (t)。如果任何值為 false，此函數會傳回 `false` (f)。

語法

```
BOOL_AND ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須為 BOOLEAN 或整數資料類型。函數的傳回類型為 BOOLEAN。

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算結果之前，將消除指定之表達式的所有重複值。如果指定引數 ALL，則函數會保留所有重複值。ALL 為預設值。如需詳細資訊，請參閱 位元彙整的 DISTINCT 支援。

範例

您可以對布林值表達式或整數表達式使用布林值函數。例如，下列查詢從 TICKET 資料庫中的標準 USERS 資料表 (其中有幾個布林值欄) 傳回結果。

BOOL_AND 函數在全部五列中傳回 `false`。其中每個州並非所有使用者都喜歡運動。

```
select state, bool_and(likesports) from users
group by state order by state limit 5;

state | bool_and
------+---------
AB    | f
AK    | f
AL    | f
```

```
AZ    | f
BC    | f
(5 rows)
```

BOOL_OR 函數

BOOL_OR 函數會對單一布林值或整數欄或表達式執行操作。此函數會將類似邏輯套用至 BIT_AND 和 BIT_OR 函數。此函數的傳回類型為布林值 (true、false 或 NULL)。

如果集合中的一個或多個值是true，BOOL_OR 函數返回true（ ）。t如果集合中的所有值都是false，函數返回false（ f ）。如果該值未知，則可以傳回 NULL。

語法

```
BOOL_OR ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須為 BOOLEAN 或整數資料類型。函數的傳回類型為 BOOLEAN。

DISTINCT | ALL

如果指定引數 DISTINCT，則函數在計算結果之前，將消除指定之表達式的所有重複值。如果指定引數 ALL，則函數會保留所有重複值。ALL 為預設值。請參閱位元彙整的 DISTINCT 支援。

範例

您可以對布林值運算式或整數運算式使用布林值函數。例如，下列查詢從 TICKET 資料庫中的標準 USERS 資料表 (其中有幾個布林值欄) 傳回結果。

BOOL_OR 函數在全部五列中傳回 true。其中每個州至少有一個使用者喜歡運動。

```
select state, bool_or(likesports) from users
group by state order by state limit 5;

state | bool_or
------+--------
```

```
AB    | t
AK    | t
AL    | t
AZ    | t
BC    | t
(5 rows)
```

以下範例傳回 NULL。

```
SELECT BOOL_OR(NULL = '123')
               bool_or
------
NULL
```

條件式運算式

主題

Amazon Redshift 支援從 SQL 標準延伸的一些條件式運算式。

CASE 條件式運算式

CASE 表達式是條件式運算式，類似於其他語言中的 if/then/else 陳述式。有多個條件時會使用 CASE 來指定結果。在 SQL 運算式有效的情況下使用 CASE，例如在 SELECT 命令中。

CASE 表達式有兩種類型：簡單和搜尋。

- 在簡單 CASE 表達式中，表達式與值相比較。發現相符時，就套用 THEN 子句中指定的動作。未發現相符時，就套用 ELSE 子句中的動作。
- 在搜尋 CASE 表達式中，每一個 CASE 的評估根據為布林值表達式，而 CASE 陳述式會傳回第一個相符的 CASE。如果在 WHEN 子句之間找不到相符項目，就傳回 ELSE 子句中的動作。

語法

用來比對條件的簡單 CASE 陳述式：

```
CASE expression
  WHEN value THEN result
  [WHEN...]
  [ELSE result]
END
```

用來評估每一個條件的搜尋 CASE 陳述式：

```
CASE
  WHEN condition THEN result
  [WHEN ...]
  [ELSE result]
END
```

引數

運算式

欄名或任何有效表達式。

值

與表達式相比較的值，例如數值常數或字元字串。

result

評估表達式或布林值條件時傳回的目標值或表達式。所有結果運算式的資料類型必須轉換為單個輸出類型。

condition

評估 true 或 false 的布林值運算式。如果 condition 為真，CASE 運算式的值是遵循條件的結果，而 CASE 運算式的其餘部分則不會處理。如果 condition 為假，則評估任何後續 WHEN 子句。如果沒有 WHEN 條件結果為真，CASE 運算式的值是 ELSE 子句的結果。如果省略 ELSE 子句且沒有條件為 true，則結果為 Null。

範例

下列範例會使用範例 TICKIT 資料中的 VENUE 表格和 SALES 表格。如需詳細資訊，請參閱 範本資料庫。

在針對 VENUE 資料表的查詢中，使用簡單 CASE 表達式以 New York City 取代 Big Apple。以 other 取代其他所有城市名稱。

```
select venuecity,
  case venuecity
    when 'New York City'
    then 'Big Apple' else 'other'
  end
from venue
order by venueid desc;

venuecity        |   case
-----------------+-----------
Los Angeles      | other
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

使用搜尋 CASE 表達式以根據個別門票銷售的 PRICEPAID 值來指派群組號碼：

```
select pricepaid,
  case when pricepaid <10000 then 'group 1'
    when pricepaid >10000 then 'group 2'
    else 'group 3'
  end
from sales
order by 1 desc;

pricepaid |  case
----------+---------
12624     | group 2
10000     | group 3
10000     | group 3
9996      | group 1
9988      | group 1
...
```

DECODE 函數

DECODE 表達式將特定值取代為另一個特定值或預設值，視等式條件的結果而定。此運算相當於簡單 CASE 表達式或 IF-THEN-ELSE 陳述式的運算。

語法

```
DECODE ( expression, search, result [, search, result ]... [ ,default ] )
```

這種表達式有助於將儲存於資料表中的縮寫或代碼，取代為報告所需的有意義商業值。

參數

運算式

您要比較之值的來源，例如資料表中的欄。

search

與來源表達式相比較的目標值，例如數值或字元字串。搜尋表達式必須評估為單一固定值。您不能指定會評估為一系列值的表達式，例如 `age between 20 and 29`；對於您要取代的每一個值，您需要指定個別的搜尋/結果對。

搜尋表達式的所有實例必須是相同或相容的資料類型。expression 和 search 參數也必須相容。

result

當表達式符合搜尋值時，查詢所傳回的替換值。DECODE 表達式必須包含至少一對搜尋/結果。

結果表達式的所有實例必須是相同或相容的資料類型。result 和 default 參數也必須相容。

default

搜尋條件失敗時用於案例的選用預設值。如果您未指定預設值，DECODE 表達式會傳回 Null。

使用須知

如果 expression 值和 search 值都是 Null，則 DECODE 結果是對應的 result 值。如需這樣使用函數的相關說明，請參閱「範例」一節。

這樣使用時，DECODE 就類似於 NVL2 函數，但有些差異。如需這些差異的描述，請參閱 NVL2 使用須知。

範例

當 datetable 的 caldate 欄中存在值 `2008-06-01` 時，下列範例會以 `June 1st, 2008` 取代此值。範例以 NULL 取代其他所有 caldate 值。

```
select decode(caldate, '2008-06-01', 'June 1st, 2008')
from datetable where month='JUN' order by caldate;

case
----------------
June 1st, 2008

...
(30 rows)
```

下列範例使用 DECODE 表達式，將 CATEGORY 資料表中五個縮寫的 CATNAME 欄轉換為全名，並將此欄中的其他值轉換為 Unknown。

```
select catid, decode(catname,
'NHL', 'National Hockey League',
'MLB', 'Major League Baseball',
'MLS', 'Major League Soccer',
'NFL', 'National Football League',
'NBA', 'National Basketball Association',
'Unknown')
from category
order by catid;

catid  | case
-------+---------------------------------
1      | Major League Baseball
2      | National Hockey League
3      | National Football League
4      | National Basketball Association
5      | Major League Soccer
6      | Unknown
7      | Unknown
8      | Unknown
9      | Unknown
10     | Unknown
11     | Unknown
(11 rows)
```

使用 DECODE 表達式來尋找科羅拉多州和內華達州在 VENUESEATS 欄為 NULL 的會場；將 NULL 轉換為零。如果 VENUESEATS 欄不是 NULL，則傳回 1 做為結果。

```
select venuename, venuestate, decode(venueseats,null,0,1)
```

```
from venue
where venuestate in('NV','CO')
order by 2,3,1;

venuename                | venuestate     | case
-------------------------------+----------------+-----------
Coors Field                    | CO         |   1
Dick's Sporting Goods Park     | CO         |   1
Ellie Caulkins Opera House     | CO         |   1
INVESCO Field         | CO         |   1
Pepsi Center         | CO         |   1
Ballys Hotel         | NV         |   0
Bellagio Hotel                 | NV         |   0
Caesars Palace                 | NV         |   0
Harrahs Hotel                  | NV         |   0
Hilton Hotel                   | NV         |   0
...
(20 rows)
```

GREATEST 和 LEAST 函數

從含有任何數量的表達式清單中傳回最大值或最小值。

語法

```
GREATEST (value [, ...])
LEAST (value [, ...])
```

參數

expression_list

逗號分隔的表達式清單，例如欄名。表達式必須全部可轉換為常見資料類型。忽略清單中的 NULL 值。如果所有表達式都評估為 NULL，則結果為 NULL。

傳回值

從提供的運算式清單中傳回最大值 (對於 GREATEST) 或最小值 (對於 LEAST)。

範例

下列範例按字母順序傳回 `firstname` 或 `lastname` 的最高值。

```
select firstname, lastname, greatest(firstname,lastname) from users
where userid < 10
order by 3;

 firstname | lastname  | greatest
-----------+-----------+-----------
 Lars      | Ratliff   | Ratliff
 Reagan    | Hodge     | Reagan
 Colton    | Roy       | Roy
 Barry     | Roy       | Roy
 Tamekah   | Juarez    | Tamekah
 Rafael    | Taylor    | Taylor
 Victor    | Hernandez | Victor
 Vladimir  | Humphrey  | Vladimir
 Mufutau   | Watkins   | Watkins
(9 rows)
```

NVL 和 COALESCE 函數

傳回一系列運算式中不為 null 的第一個運算式的值。找到非 Null 值時，就不會評估清單中剩餘的運算式。

NVL 與 COALESCE 相同。它們是同義詞。本主題說明語法，並包含兩者的範例。

語法

```
NVL( expression, expression, ... )
```

COALESCE 的語法是相同的：

```
COALESCE( expression, expression, ... )
```

如果所有表達式都是 Null，則結果為 Null。

當您想要在主要值遺失或為 null 時傳回次要值，這些函數非常有用。例如，查詢可能會傳回三個可用電話號碼中的第一個：行動電話號碼、住家或公司。函數中運算式的順序決定評估的順序。

引數

運算式

要評估 Null 狀態的表達式，例如欄名。

傳回類型

Amazon Redshift 會根據輸入運算式判斷傳回值的資料類型。如果輸入運算式的資料類型沒有一般類型，則會傳回錯誤。

範例

如果清單包含整數運算式，該函數傳回一個整數。

```
SELECT COALESCE(NULL, 12, NULL);

coalesce
--------------
12
```

這個範例與前面的範例相同，不同之處在於它使用 NVL，會傳回相同的結果。

```
SELECT NVL(NULL, 12, NULL);

coalesce
--------------
12
```

下列範例會傳回字串類型。

```
SELECT COALESCE(NULL, 'Amazon Redshift', NULL);

coalesce
--------------
Amazon Redshift
```

下列範例會導致錯誤，因為運算式清單中的資料類型不同。在這種情況下，清單中同時存在字串類型和數字類型。

```
SELECT COALESCE(NULL, 'Amazon Redshift', 12);
ERROR: invalid input syntax for integer: "Amazon Redshift"
```

在此範例中，您建立具有 START_DATE 和 END_DATE 欄的資料表、插入包含 Null 值的列，然後將 NVL 運算式套用至這兩欄。

```
create table datetable (start_date date, end_date date);
```

```
insert into datetable values ('2008-06-01','2008-12-31');
insert into datetable values (null,'2008-12-31');
insert into datetable values ('2008-12-31',null);
```

```
select nvl(start_date, end_date)
from datetable
order by 1;

coalesce
------------
2008-06-01
2008-12-31
2008-12-31
```

NVL 表達式的預設欄名為 COALESCE。下列查詢會傳回相同的結果：

```
select coalesce(start_date, end_date)
from datetable
order by 1;
```

對於下列範例查詢，您可以建立包含飯店預訂範例資訊的表格，並插入數列。有些記錄包含 null 值。

```
create table booking_info (booking_id int, booking_code character(8), check_in date,
 check_out date, funds_collected numeric(12,2));
```

插入下列範例資料。有些記錄沒有 check_out 日期或 funds_collected 金額。

```
insert into booking_info values (1, 'OCEAN_WV', '2023-02-01','2023-02-03',100.00);
insert into booking_info values (2, 'OCEAN_WV', '2023-04-22','2023-04-26',120.00);
insert into booking_info values (3, 'DSRT_SUN', '2023-03-13','2023-03-16',125.00);
insert into booking_info values (4, 'DSRT_SUN', '2023-06-01','2023-06-03',140.00);
insert into booking_info values (5, 'DSRT_SUN', '2023-07-10',null,null);
insert into booking_info values (6, 'OCEAN_WV', '2023-08-15',null,null);
```

下列查詢會傳回日期清單。如果 check_out 日期不可用，它會列出 check_in 日期。

```
select coalesce(check_out, check_in)
from booking_info
order by booking_id;
```

結果如下。請注意，最後兩則記錄顯示 check_in 日期。

```
coalesce
------------
2023-02-03
2023-04-26
2023-03-16
2023-06-03
2023-07-10
2023-08-15
```

如果您預期查詢的某些函數或欄會傳回 Null 值，您可以使用 NVL 表達式以其他值取代 Null。例如，彙總函數 (例如 SUM) 在沒有可評估的列時會傳回 Null 值，而不是零。您可以使用 NVL 運算式，以 700.0 取代這些 Null 值：加總 funds_collected 的結果不是 485 而是 1885，因為兩個有 null 的列被替換為 700。

```
select sum(nvl(funds_collected, 700.0)) as sumresult from booking_info;

sumresult
------
 1885
```

NVL2 函數

根據指定的表達式評估為 NULL 還是 NOT NULL，傳回兩個值中的一個。

語法

```
NVL2 ( expression, not_null_return_value, null_return_value )
```

引數

運算式

要評估 Null 狀態的表達式，例如欄名。

not_null_return_value

expression 評估為 NOT NULL 時所傳回的值。not_null_return_value 值的資料類型必須與 expression 相同，或可隱含地轉換為該資料類型。

null_return_value

expression 評估為 NULL 時所傳回的值。null_return_value 值的資料類型必須與 expression 相同，或可隱含地轉換為該資料類型。

傳回類型

NVL2 傳回類型的決定方式如下：

- 如果 not_null_return_value 或 null_return_value 為 Null，則傳回 not-null 表達式的資料類型。

如果 not_null_return_value 和 null_return_value 都不是 Null：

- 如果 not_null_return_value 和 null_return_value 有相同的資料類型，則傳回該資料類型。
- 如果 not_null_return_value 和 null_return_value 有不同的數值資料類型，則傳回最小可相容的數值資料類型。
- 如果 not_null_return_value 和 null_return_value 有不同的日期時間資料類型，則傳回時間戳記資料類型。
- 如果 not_null_return_value 和 null_return_value 有不同的字元資料類型，則傳回 not_null_return_value 的資料類型。
- 如果 not_null_return_value 和 null_return_value 有混合的數值和非數值資料類型，則傳回 not_null_return_value 的資料類型。

> **Important**
>
> 前兩個案例中傳回 not_null_return_value 的資料類型，而 null_return_value 會隱含地轉換為該資料類型。如果資料類型不相容，函數會失敗。

使用須知

當 expression 和 search 參數都為 null 時，可以用類似於 NVL2 的方式來使用 DECODE 函數。差別在於 DECODE 會同時傳回 result 參數的值和資料類型。反之，NVL2 會傳回 not_null_return_value 或 null_return_value 參數的值 (視函數選取何者而定)，但傳回值會有 not_null_return_value 的資料類型。

例如，假設 column1 是 NULL，下列查詢會傳回相同的值。不過，DECODE 傳回值的資料類型為 INTEGER，而 NVL2 傳回值的資料類型為 VARCHAR。

```
select decode(column1, null, 1234, '2345');
select nvl2(column1, '2345', 1234);
```

範例

下列範例修改一些範例資料，然後評估兩個欄位來提供使用者的適當聯絡資訊：

```
update users set email = null where firstname = 'Aphrodite' and lastname = 'Acevedo';

select (firstname + ' ' + lastname) as name,
nvl2(email, email, phone) AS contact_info
from users
where state = 'WA'
and lastname  like 'A%'
order by lastname, firstname;

name        contact_info
--------------------+--------------------------------------------
Aphrodite Acevedo (906) 632-4407
Caldwell Acevedo  Nunc.sollicitudin@Duisac.ca
Quinn Adams    vel@adipiscingligulaAenean.com
Kamal Aguilar  quis@vulputaterisusa.com
Samson Alexander  hendrerit.neque@indolorFusce.ca
Hall Alford    ac.mattis@vitaediamProin.edu
Lane Allen     et.netus@risusDonec.org
Xander Allison    ac.facilisis.facilisis@Infaucibus.com
Amaya Alvarado    dui.nec.tempus@eudui.edu
Vera Alvarez   at.arcu.Vestibulum@pellentesque.edu
Yetta Anthony  enim.sit@risus.org
Violet Arnold  ad.litora@at.com
August Ashley  consectetuer.euismod@Phasellus.com
Karyn Austin   ipsum.primis.in@Maurisblanditenim.org
Lucas Ayers    at@elitpretiumet.com
```

NULLIF 函數

語法

NULLIF 表達式比較兩個引數，如果引數相等，則傳回 Null。如果不相等，則傳回第一個引數。此表達式與 NVL 或 COALESCE 表達式相反。

```
NULLIF ( expression1, expression2 )
```

引數

expression1、expression2

比較的目標欄或表達式。傳回類型與第一個表達式的類型相同。NULLIF 結果的預設欄名為第一個表達式的欄名。

範例

在下列範例中，查詢會傳回字串 first，因為引數不相等。

```
SELECT NULLIF('first', 'second');

case
-------
first
```

在下列範例中，查詢會傳回 NULL，因為字串常值引數相等。

```
SELECT NULLIF('first', 'first');

case
-------
NULL
```

在下列範例中，查詢會傳回 1，因為整數引數不相等。

```
SELECT NULLIF(1, 2);

case
-------
1
```

在下列範例中，查詢會傳回 NULL，因為整數引數相等。

```
SELECT NULLIF(1, 1);

case
-------
NULL
```

在下列範例中，當 LISTID 和 SALESID 值相符時，查詢會傳回 Null：

```
select nullif(listid,salesid), salesid
from sales where salesid<10 order by 1, 2 desc;

listid  | salesid
--------+---------
     4  |       2
     5  |       4
     5  |       3
     6  |       5
     10 |       9
     10 |       8
     10 |       7
     10 |       6
        |       1
(9 rows)
```

您可以使用 NULLIF 來確保空字串一律以 Null 傳回。在下列範例中，NULLIF 表達式傳回 Null 值或至少包含一個字元的字串。

```
insert into category
values(0,'','Special','Special');

select nullif(catgroup,'') from category
where catdesc='Special';

catgroup
----------
null
(1 row)
```

NULLIF 會忽略結尾空格。如果字串不是空的但包含空格，NULLIF 仍會傳回 Null：

```
create table nulliftest(c1 char(2), c2 char(2));

insert into nulliftest values ('a','a ');

insert into nulliftest values ('b','b');


select nullif(c1,c2) from nulliftest;
```

```
c1
------
null
null
(2 rows)
```

資料類型格式化函數

主題

- CAST 函數
- CONVERT 函數
- TO_CHAR
- TO_DATE 陣列
- TO_NUMBER
- TEXT_TO_INT_ALT
- TEXT_TO_NUMERIC_ALT
- 日期時間格式字串
- 數值格式字串
- 數值資料的 Teradata 樣式格式化字元

資料類型格式化函數可讓您輕鬆轉換值的資料類型。範本對於其中每一個函數，第一個引數一律為要格式化的值，第二個引數包含新格式的範本。Amazon Redshift 支援多種資料類型格式化函數。

CAST 函數

CAST 函數會將一種資料類型轉換為另一個相容的資料類型。例如，您可以將字串轉換為日期，或將數值類型轉換為字串。CAST 會執行執行期轉換，這表示轉換不會變更來源資料表中值的資料類型。它僅在查詢的上下文中進行更改。

CAST 函數非常類似 the section called “CONVERT”，因為它們都將一種資料類型轉換為另一種資料類型，但它們的呼叫方式不同。

某些資料類型需要使用 CAST 或 CONVERT 函數來明確轉換為其他資料類型。其他資料類型可在另一個命令中隱含地轉換，而不需要使用 CAST 或 CONVERT。請參閱類型相容性與轉換。

語法

使用兩種同等的語法格式中的任何一種，將運算式從一種資料類型轉換為另一種資料類型。

```
CAST ( expression AS type )
expression :: type
```

引數

運算式

任何評估為一或多個值的表達式，例如欄名或常值。轉換 Null 值會傳回 Null。表達式不能包含空格或空字串。

type

其中一個支援的資料類型。

傳回類型

CAST 傳回 type 引數指定的資料類型。

> **Note**
>
> 如果您嘗試執行有問題的轉換，例如 DECIMAL 轉換導致精確度降低，Amazon Redshift 會傳回錯誤：
>
> ```
> select 123.456::decimal(2,1);
> ```
>
> 或造成溢位的 INTEGER 轉換：
>
> ```
> select 12345678::smallint;
> ```

範例

一些範例使用範例 TICKIT 資料庫。如需有關設定範例資料的詳細資訊，請參閱載入資料。

下列兩個查詢相同。都是將小數值轉換為整數：

```
select cast(pricepaid as integer)
from sales where salesid=100;

pricepaid
```

```
-----------
162
(1 row)
```

```
select pricepaid::integer
from sales where salesid=100;

pricepaid
-----------
162
(1 row)
```

以下產生類似的結果。它不需要執行範例資料：

```
select cast(162.00 as integer) as pricepaid;

pricepaid
-----------
162
(1 row)
```

在此範例中，時間戳記資料欄中的值會轉換為日期，因此會從每個結果中移除時間：

```
select cast(saletime as date), salesid
from sales order by salesid limit 10;

 saletime  | salesid
-----------+---------
2008-02-18 |       1
2008-06-06 |       2
2008-06-06 |       3
2008-06-09 |       4
2008-08-31 |       5
2008-07-16 |       6
2008-06-26 |       7
2008-07-10 |       8
2008-07-22 |       9
2008-08-06 |      10
(10 rows)
```

如果您沒有按照上一個範例中所示使用 CAST，則結果將包括時間：2008-02-18 02:36:48。

下列查詢會將可變字元資料轉換為日期。它不需要執行範例資料。

```
select cast('2008-02-18 02:36:48' as date) as mysaletime;

mysaletime
--------------------
2008-02-18
(1 row)
```

在此範例中，日期欄中的值轉換為時間戳記：

```
select cast(caldate as timestamp), dateid
from date order by dateid limit 10;

      caldate       | dateid
--------------------+--------
2008-01-01 00:00:00 |   1827
2008-01-02 00:00:00 |   1828
2008-01-03 00:00:00 |   1829
2008-01-04 00:00:00 |   1830
2008-01-05 00:00:00 |   1831
2008-01-06 00:00:00 |   1832
2008-01-07 00:00:00 |   1833
2008-01-08 00:00:00 |   1834
2008-01-09 00:00:00 |   1835
2008-01-10 00:00:00 |   1836
(10 rows)
```

在類似前面範例的情況下，您可以使用 TO_CHAR 來獲得輸出格式的額外控制。

在此範例中，整數轉換為字元字串：

```
select cast(2008 as char(4));

bpchar
--------
2008
```

在此範例中，DECIMAL(6,3) 值轉換為 DECIMAL(4,1) 值：

```
select cast(109.652 as decimal(4,1));
```

```
numeric
---------
109.7
```

此範例顯示更複雜的運算式。SALES 資料表中的 PRICEPAID 欄 (DECIMAL(8,2) 欄) 轉換為 DECIMAL(38,2) 欄，並將值乘以 100000000000000000000：

```
select salesid, pricepaid::decimal(38,2)*100000000000000000000
as value from sales where salesid<10 order by salesid;


 salesid |           value
---------+----------------------------
       1 | 728000000000000000000000.00
       2 |  760000000000000000000000.00
       3 | 350000000000000000000000.00
       4 | 175000000000000000000000.00
       5 | 154000000000000000000000.00
       6 | 394000000000000000000000.00
       7 | 788000000000000000000000.00
       8 | 197000000000000000000000.00
       9 | 591000000000000000000000.00
(9 rows)
```

> **Note**
>
> 您無法在 GEOMETRY 資料類型上執行 CAST 或 CONVERT 操作，將其變更為其他資料類型。但是，您可以為接受 GEOMETRY 引數的函數提供擴充已知二進位 (EWKB) 格式的字串常值十六進位表示法做為輸入。例如，以下 ST_AsText 函數預期 GEOMETRY 資料類型。
>
> ```
> SELECT ST_AsText('010100000000000000000001C40000000000002040');
> ```
>
> ```
> st_astext
> ------------
>  POINT(7 8)
> ```
>
> 您也可以明確指定 GEOMETRY 資料類型。
>
> ```
> SELECT ST_AsText('010100000000000000000001440000000000001840'::geometry);
> ```

```
st_astext
------------
 POINT(5 6)
```

CONVERT 函數

與 CAST 函數一樣，CONVERT 函數可將一種資料類型轉換為另一個相容的資料類型。例如，您可以將字串轉換為日期，或將數字類型轉換為字串。CONVERT 會執行執行期轉換，這表示轉換不會變更來源資料表中值的資料類型。它僅在查詢的上下文中進行更改。

某些資料類型需要使用 CONVERT 函數來明確轉換為其他資料類型。其他資料類型可在另一個命令中隱含地轉換，而不需要使用 CAST 或 CONVERT。請參閱類型相容性與轉換。

語法

```
CONVERT ( type, expression )
```

引數

type

其中一個支援的資料類型。

運算式

任何評估為一或多個值的表達式，例如欄名或常值。轉換 Null 值會傳回 Null。表達式不能包含空格或空字串。

傳回類型

CONVERT 傳回 type 引數指定的資料類型。

> **Note**
>
> 如果您嘗試執行有問題的轉換，例如 DECIMAL 轉換導致精確度降低，Amazon Redshift 會傳回錯誤：
>
> ```
> SELECT CONVERT(decimal(2,1), 123.456);
> ```

或造成溢位的 INTEGER 轉換：

```
SELECT CONVERT(smallint, 12345678);
```

範例

一些範例使用範例 TICKIT 資料庫。如需有關設定範例資料的詳細資訊，請參閱載入資料。

下列查詢使用 CONVERT 函數將一欄小數轉換為整數

```
SELECT CONVERT(integer, pricepaid)
FROM sales WHERE salesid=100;
```

這個範例會將整數轉換為字元字串。

```
SELECT CONVERT(char(4), 2008);
```

此範例會將目前日期和時間轉換為可變字元資料類型：

```
SELECT CONVERT(VARCHAR(30), GETDATE());

getdate
---------
2023-02-02 04:31:16
```

這個範例會將 saletime 欄轉換成僅限時間，並從每一列移除日期。

```
SELECT CONVERT(time, saletime), salesid
FROM sales order by salesid limit 10;
```

如需將時間戳記從某個時區轉換為另一個時區的詳細資訊，請參閱CONVERT_TIMEZONE 函數。如需其他日期和時間函數，請參閱日期和時間函數。

下列範例會將可變字元資料轉換成日期時間物件。

```
SELECT CONVERT(datetime, '2008-02-18 02:36:48') as mysaletime;
```

> **Note**
>
> 您無法在 GEOMETRY 資料類型上執行 CAST 或 CONVERT 操作，將其變更為其他資料類型。但是，您可以為接受 GEOMETRY 引數的函數提供擴充已知二進位 (EWKB) 格式的字串常值十六進位表示法做為輸入。例如，以下 ST_AsText 函數預期 GEOMETRY 資料類型。
>
> ```
> SELECT ST_AsText('010100000000000000000001C40000000000002040');
> ```
>
> ```
> st_astext
> ------------
>  POINT(7 8)
> ```
>
> 您也可以明確指定 GEOMETRY 資料類型。
>
> ```
> SELECT ST_AsText('010100000000000000000014400000000000001840'::geometry);
> ```
>
> ```
> st_astext
> ------------
>  POINT(5 6)
> ```

TO_CHAR

TO_CHAR 將時間戳記或數值運算式轉換為字元字串資料格式。

語法

```
TO_CHAR (timestamp_expression | numeric_expression , 'format')
```

引數

timestamp_expression

此運算式產生 TIMESTAMP 或 TIMESTAMPTZ 類型值，或可隱含地強制轉換為時間戳記的值。

numeric_expression

此表達式產生數值資料類型的值，或可隱含地強制轉換為數值類型的值。如需詳細資訊，請參閱 數值類型。TO_CHAR 在數值字串左側插入空格。

> **Note**
>
> TO_CHAR 不支援 128 位元 DECIMAL 值。

format

新值的格式。關於有效的格式，請參閱日期時間格式字串和 數值格式字串。

傳回類型

VARCHAR

範例

下列範例會將時間戳記轉換為日期和時間的值，其格式為月份名稱填滿九個字元、星期名稱以及月份的日期編號。

```
select to_char(timestamp '2009-12-31 23:15:59', 'MONTH-DY-DD-YYYY HH12:MIPM');

to_char
-------------------------
DECEMBER -THU-31-2009 11:15PM
```

下列範例會將時間戳記轉換為具有年份天數的值。

```
select to_char(timestamp '2009-12-31 23:15:59', 'DDD');

to_char
-------------------------
365
```

下列範例會將時間戳記轉換為一週的 ISO 天數。

```
select to_char(timestamp '2022-05-16 23:15:59', 'ID');

to_char
-------------------------
1
```

以下範例會從日期擷取月份名稱。

```
select to_char(date '2009-12-31', 'MONTH');

to_char
-------------------------
DECEMBER
```

下列範例將 EVENT 資料表中的每一個 STARTTIME 值，轉換為由時、分、秒組成的字串。

```
select to_char(starttime, 'HH12:MI:SS')
from event where eventid between 1 and 5
order by eventid;

to_char
----------
02:30:00
08:00:00
02:30:00
02:30:00
07:00:00
```

下列範例將整個時間戳記值轉換成另一種格式。

```
select starttime, to_char(starttime, 'MON-DD-YYYY HH12:MIPM')
from event where eventid=1;

      starttime      |       to_char
---------------------+---------------------
 2008-01-25 14:30:00 | JAN-25-2008 02:30PM
```

下列範例將時間戳記常值轉換為字元字串。

```
select to_char(timestamp '2009-12-31 23:15:59','HH24:MI:SS');

to_char
----------
23:15:59
```

下列範例會將十進位數字轉換為字元字串。

```
select to_char(125.8, '999.99');
```

```
to_char
---------
125.80
```

下列範例會將十進位數字轉換為字元字串。

```
select to_char(125.8, '999D99');

to_char
---------
125.80
```

下列範例會將數字轉換為前導零的字元字串。

```
select to_char(125.8, '0999D99');

to_char
---------
0125.80
```

下列範例將數字轉換為結尾帶負號的字元字串。

```
select to_char(-125.8, '999D99S');

to_char
---------
125.80-
```

下列範例會將數字轉換為字元字串，且正號或負號位於指定位置。

```
select to_char(125.8, '999D99SG');

to_char
---------
125.80+
```

下列範例會將數字轉換為字元字串，且正號位於指定位置。

```
select to_char(125.8, 'PL999D99');

to_char
```

```
---------
+ 125.80
```

下列範例將數字轉換為帶有貨幣符號的字元字串。

```
select to_char(-125.88, '$S999D99');

to_char
---------
$-125.88
```

下列範例會將數字轉換為字元字串，且貨幣符號位於指定位置。

```
select to_char(-125.88, 'S999D99L');

to_char
---------
-125.88$
```

下列範例會使用千位 (逗號) 分隔符號，將數字轉換為字元字串。

```
select to_char(1125.8, '9,999.99');

to_char
---------
1,125.80
```

下列範例將數字轉換為字元字串，並使用角括號代表負數。

```
select to_char(-125.88, '$999D99PR');

to_char
---------
$<125.88>
```

下列範例將數字轉換為 Roman 數值字串。

```
select to_char(125, 'RN');

to_char
---------
```

```
   CXXV
```

下面的例子將日期轉換為世紀代碼。

```
select to_char(date '2020-12-31', 'CC');

to_char
---------
21
```

下列範例會顯示星期幾。

```
SELECT to_char(current_timestamp, 'FMDay, FMDD HH12:MI:SS');

to_char
-----------------------
Wednesday, 31 09:34:26
```

下列範例顯示數字的序號字尾。

```
SELECT to_char(482, '999th');

to_char
-----------------------
 482nd
```

下列範例將 sales 資料表中的支付價格減去佣金。然後，將差額捨進並轉換為羅馬數字，並顯示於 to_char 欄：

```
select salesid, pricepaid, commission, (pricepaid - commission)
as difference, to_char(pricepaid - commission, 'rn') from sales
group by sales.pricepaid, sales.commission, salesid
order by salesid limit 10;

 salesid | pricepaid | commission | difference |     to_char
---------+-----------+------------+------------+-----------------
       1 |    728.00 |     109.20 |     618.80 |           dcxix
       2 |     76.00 |      11.40 |      64.60 |             lxv
       3 |    350.00 |      52.50 |     297.50 |        ccxcviii
       4 |    175.00 |      26.25 |     148.75 |           cxlix
       5 |    154.00 |      23.10 |     130.90 |           cxxxi
       6 |    394.00 |      59.10 |     334.90 |         cccxxxv
```

```
       7 |    788.00 |     118.20 |     669.80 |           dclxx
       8 |    197.00 |      29.55 |     167.45 |          clxvii
       9 |    591.00 |      88.65 |     502.35 |             dii
      10 |     65.00 |       9.75 |      55.25 |              lv
```

下列範例將 to_char 欄所顯示的差額值加上貨幣符號：

```
select salesid, pricepaid, commission, (pricepaid - commission)
as difference, to_char(pricepaid - commission, 'l99999D99') from sales
group by sales.pricepaid, sales.commission, salesid
order by salesid limit 10;

salesid | pricepaid | commission | difference |  to_char
--------+-----------+------------+------------+------------
      1 |    728.00 |     109.20 |     618.80 | $   618.80
      2 |     76.00 |      11.40 |      64.60 | $    64.60
      3 |    350.00 |      52.50 |     297.50 | $   297.50
      4 |    175.00 |      26.25 |     148.75 | $   148.75
      5 |    154.00 |      23.10 |     130.90 | $   130.90
      6 |    394.00 |      59.10 |     334.90 | $   334.90
      7 |    788.00 |     118.20 |     669.80 | $   669.80
      8 |    197.00 |      29.55 |     167.45 | $   167.45
      9 |    591.00 |      88.65 |     502.35 | $   502.35
     10 |     65.00 |       9.75 |      55.25 | $    55.25
```

下列範例列出每一筆銷售完成的世紀。

```
select salesid, saletime, to_char(saletime, 'cc') from sales
order by salesid limit 10;

 salesid |      saletime       | to_char
---------+---------------------+---------
       1 | 2008-02-18 02:36:48 | 21
       2 | 2008-06-06 05:00:16 | 21
       3 | 2008-06-06 08:26:17 | 21
       4 | 2008-06-09 08:38:52 | 21
       5 | 2008-08-31 09:17:02 | 21
       6 | 2008-07-16 11:59:24 | 21
       7 | 2008-06-26 12:56:06 | 21
       8 | 2008-07-10 02:12:36 | 21
       9 | 2008-07-22 02:23:17 | 21
      10 | 2008-08-06 02:51:55 | 21
```

下列範例將 EVENT 資料表中的每一個 STARTTIME 值，轉換為由時、分、秒及時區組成的字串。

```
select to_char(starttime, 'HH12:MI:SS TZ')
from event where eventid between 1 and 5
order by eventid;

to_char
----------
02:30:00 UTC
08:00:00 UTC
02:30:00 UTC
02:30:00 UTC
07:00:00 UTC
```

下列範例顯示秒、毫秒和微秒的格式。

```
select sysdate,
to_char(sysdate, 'HH24:MI:SS') as seconds,
to_char(sysdate, 'HH24:MI:SS.MS') as milliseconds,
to_char(sysdate, 'HH24:MI:SS:US') as microseconds;

timestamp           | seconds  | milliseconds | microseconds
--------------------+----------+--------------+----------------
2015-04-10 18:45:09 | 18:45:09 | 18:45:09.325 | 18:45:09:325143
```

TO_DATE 陣列

TO_DATE 將字元字串所表示的日期轉換為 DATE 資料類型。

語法

```
TO_DATE(string, format)
```

```
TO_DATE(string, format, is_strict)
```

引數

string

要轉換的字串。

format

字串常值，定義輸出 string 的日期部分格式。如需有效日、月和年格式的清單，請參閱日期時間格式字串。

is_strict

選用的布林值，指定如果輸入日期值超出範圍，是否會傳回錯誤。當 is_strict 設定為 TRUE 時，如果有超出範圍的值，就會傳回錯誤。當 is_strict 設定為 FALSE (預設值) 時，就會接受溢位值。

傳回類型

TO_DATE 傳回 DATE，視 format 值而定。

如果轉換成 format 失敗，則會傳回錯誤。

範例

下列 SQL 陳述式會將日期 02 Oct 2001 轉換為日期資料類型。

```
select to_date('02 Oct 2001', 'DD Mon YYYY');

to_date
------------
2001-10-02
(1 row)
```

下列 SQL 陳述式會將字串 20010631 轉換為日期。

```
select to_date('20010631', 'YYYYMMDD', FALSE);
```

結果是 2001 年 7 月 1 日，因為 6 月只有 30 天。

```
to_date
------------
2001-07-01
```

下列 SQL 陳述式會將字串 20010631 轉換為日期：

```
to_date('20010631', 'YYYYMMDD', TRUE);
```

結果是錯誤，因為六月只有 30 天。

```
ERROR:  date/time field date value out of range: 2001-6-31
```

TO_NUMBER

TO_NUMBER 將字串轉換為數值 (十進位)。

語法

```
to_number(string, format)
```

引數

string

要轉換的字串。格式必須是文字值。

format

第二個引數是格式字串，指出如何剖析字元字串來建立數值。例如，格式 '99D999' 指定要轉換的字串包含五位數，且第三個位置是小數點。例如，to_number('12.345','99D999') 會將以數值傳回 12.345。如需有效格式的清單，請參閱 數值格式字串。

傳回類型

TO_NUMBER 傳回 DECIMAL 數字。

如果轉換成 format 失敗，則會傳回錯誤。

範例

下列範例將字串 12,454.8- 轉換為數字：

```
select to_number('12,454.8-', '99G999D9S');

to_number
-----------
-12454.8
```

下列範例將字串 $ 12,454.88 轉換為數字：

```
select to_number('$ 12,454.88', 'L 99G999D99');

to_number
-----------
12454.88
```

下列範例將字串 $ 2,012,454.88 轉換為數字：

```
select to_number('$ 2,012,454.88', 'L 9,999,999.99');

to_number
-----------
2012454.88
```

TEXT_TO_INT_ALT

TEXT_TO_INT_ALT 會將字元字串轉換為使用 Teradata 樣式格式的整數。結果中的分數數字會被截斷。

語法

```
TEXT_TO_INT_ALT (expression [ , 'format'])
```

引數

運算式

產生一個或多個 CHAR 或 VARCHAR 值的運算式，例如欄名稱或常值字串。轉換 Null 值會傳回 Null。該函數將空白或空字串轉換為 0。

format

字串常值，定義輸入運算式的格式。如需您可以指定之格式化字元的相關資訊，請參閱數值資料的 Teradata 樣式格式化字元。

傳回類型

TEXT_TO_INT_ALT 會傳回 INTEGER 值。

轉換結果的小數部分被截斷。

如果轉換成您指定的 format 詞語不成功，Amazon Redshift 就會傳回錯誤訊息。

範例

下列範例會將輸入 expression 字串 '123-' 轉換為整數 -123。

```
select text_to_int_alt('123-');
```

```
text_to_int_alt
----------
      -123
```

下列範例會將輸入 expression 字串 '2147483647+' 轉換成整數 2147483647。

```
select text_to_int_alt('2147483647+');
```

```
text_to_int_alt
----------
2147483647
```

下列範例會將指數輸入 expression 字串 '-123E-2' 轉換為整數 -1。

```
select text_to_int_alt('-123E-2');
```

```
text_to_int_alt
----------
        -1
```

下列範例會將輸入 expression 字串 '2147483647+' 轉換成整數 2147483647。

```
select text_to_int_alt('2147483647+');
```

```
text_to_int_alt
----------
2147483647
```

以下範例會將 format 詞語 '999S' 的輸入 expression 字串 '123{' 轉換為整數 1230。S 字元表示帶符號的分區十進位值。如需詳細資訊，請參閱 數值資料的 Teradata 樣式格式化字元。

```
text_to_int_alt('123{', '999S');
```

```
text_to_int_alt
----------
      1230
```

以下範例會將 format 詞語 'C9(I)' 的輸入 expression 字串 'USD123' 轉換為整數 123。請參閱數值資料的 Teradata 樣式格式化字元。

```
text_to_int_alt('USD123', 'C9(I)');
```

```
text_to_int_alt
----------
       123
```

下列範例會指定資料表欄做為輸入 expression。

```
select text_to_int_alt(a), text_to_int_alt(b) from t_text2int order by 1;
```

```
 text_to_int_alt | text_to_int_alt
-----------------+-----------------
            -123 |            -123
            -123 |            -123
             123 |             123
             123 |             123
```

以下是此範例的表格定義和插入陳述式。

```
create table t_text2int (a varchar(200), b char(200));
```

```
insert into t_text2int VALUES('123', '123'),('123.123', '123.123'), ('-123', '-123'),
 ('123-', '123-');
```

TEXT_TO_NUMERIC_ALT

TEXT_TO_NUMERIC_ALT 會執行 Teradata 樣式的轉換操作，將字元字串轉換為數字資料格式。

語法

```
TEXT_TO_NUMERIC_ALT (expression [, 'format'] [, precision, scale])
```

引數

運算式

任何評估為一或多個 CHAR 或 VARCHAR 值的運算式，例如欄名稱或常值。轉換 Null 值會傳回 Null。空白或空字串會轉換為 0。

format

字串常值，定義輸入運算式的格式。如需詳細資訊，請參閱 數值資料的 Teradata 樣式格式化字元。

precision

數字結果中的位數。預設值為 38。

scale

數值結果中小數點右邊的位數。預設值為 0。

傳回類型

TEXT_TO_NUMERIC_ALT 會傳回 DECIMAL 數字。

如果轉換成您指定的 format 詞語不成功，Amazon Redshift 就會傳回錯誤訊息。

Amazon Redshift 會將輸入 expression 字串轉換為數值類型，並以您在精確度選項中為該類型指定的最高 precision。如果數值的長度超過您為 precision 指定的值，Amazon Redshift 會根據下列規則四捨五入數值：

- 如果轉換結果的長度超過您在 format 詞語中指定的長度，Amazon Redshift 會傳回錯誤訊息。
- 如果將結果轉換為數值，則結果會四捨五入為最接近的值。如果小數部分剛好位於上部和下部轉換結果的中間，則結果將四捨五入到最接近的偶數。

範例

下列範例會將輸入 expression 字串 '1.5' 轉換為數值 '2'。因為陳述式未指定 scale，所以 scale 預設為 0，而轉換結果不包含分數結果。由於 0.5 位於 1 和 2 之間，因此轉換結果將四捨五入為偶數值 2。

```
select text_to_numeric_alt('1.5');
```

```
 text_to_numeric_alt
---------------------
```

```
                    2
```

下列範例會將輸入 expression 字串 '2.51' 轉換為數值 3。因為陳述式未指定 scale 值，所以 scale 預設為 0，而轉換結果不包含分數結果。由於 0.51 比 2 更接近 3，因此轉換結果將四捨五入為值 3。

```
select text_to_numeric_alt('2.51');
```

```
 text_to_numeric_alt
---------------------
                    3
```

下列範例會將 precision 為 10 且 scale 為 2 的輸入 expression 字串 123.52501 轉換為數值 123.53。

```
select text_to_numeric_alt('123.52501', 10, 2);
```

```
 text_to_numeric_alt
---------------------
              123.53
```

下列範例會指定資料表欄做為輸入 expression。

```
select text_to_numeric_alt(a), text_to_numeric_alt(b) from t_text2numeric order by 1;
```

```
           text_to_numeric_alt           |           text_to_numeric_alt
-----------------------------------------+-----------------------------------------
 -99999999999999999999999999999999999999 | -99999999999999999999999999999999999999
                                  -12300 |                                  -12300
                                     123 |                                     123
                                     123 |                                     123
  99999999999999999999999999999999999999 |  99999999999999999999999999999999999999
```

以下是此範例的表格定義和插入陳述式。

```
create table  t_text2numeric (a varchar(200), b char(200));
```

```
insert into  t_text2numeric values
('123', '123'),
('+123.456', '+123.456'),
('-' || repeat('9', 38), '-' || repeat('9', 38)),
```

```
(repeat('9', 38) || '+', repeat('9', 38) || '+'),
('-123E2', '-123E2');
```

日期時間格式字串

以下提供日期時間格式字串的參考。

下列格式字串適用於 TO_CHAR 之類的函數。這些字串可以包含日期時間分隔符號 (例如 '-'、'/' 或 ':') 及下列「日期部分」和「時間部分」。

日期部分或時間部分	意義
BC 或 B.C.、AD 或 A.D.、b.c. 或 bc、ad 或 a.d。	大寫和小寫的紀元標記
CC	兩位數世紀編號
YYYY、YYY、YY、Y	4 位數、3 位數、2 位數、1 位數的年編號
Y、YYY	含逗號的 4 位數年編號
IYYY、IYY、IY、I	4 位數、3 位數、2 位數、1 位數的國際標準組織 (ISO) 年編號
Q	季編號 (1 到 4)
MONTH、Month、month	月名稱 (大寫、大小寫混合、小寫、以空格填補為 9 個字元)
MON、Mon、mon	縮寫月名稱 (大寫、大小寫混合、小寫、以空格填補為 3 個字元)
MM	月編號 (01-12)
RM、rm	羅馬數字的月編號 (I-XII，其中 I 代表一月，大寫或小寫)
W	月的第幾週 (1-5；第一週以月初第一天起算。)
WW	年的週編號 (1-53；第一週以年初第一天起算。)

日期部分或時間部分	意義
IW	年的 ISO 週編號 (新年第一個星期四在第 1 週。)
DAY、Day、day	日名稱 (大寫、大小寫混合、小寫、以空格填補為 9 個字元)
DY、Dy、dy	縮寫日名稱 (大寫、大小寫混合、小寫、以空格填補為 3 個字元)
DDD	年的第幾日 (001-366)
IDDD	ISO 8601 週編號年的第幾日 (001-371；一年的第 1 日是第一個 ISO 週的星期一)
DD	月的第幾日，以數字表示 (01-31)
D	星期幾 (1-7；星期日是 1) **Note** D 日期部分的行為不同於日期時間函數 DATE_PART 和 EXTRACT 使用的星期幾 (DOW) 日期部分。DOW 是基於整數 0-6，其中星期日為 0。如需詳細資訊，請參閱 日期或時間戳記函數的日期部分。
ID	ISO 8601 一週的星期幾，星期一 (1) 至星期日 (7)
J	羅馬曆日 (自紀元前 4712 年 1 月 1 日起算的天數)
HH24	小時 (24 小時制，00-23)
HH 或 HH12	小時 (12 小時制，01-12)

日期部分或時間部分	意義
MI	分鐘 (00–59)
SS	秒 (00–59)
MS	毫秒 (.000)
US	微秒 (.000000)
AM 或 PM、A.M. 或 P.M.、a.m. 或 p.m.、am 或 pm	大寫和小寫正午指標 (用於 12 小時制)
TZ、tz	大寫和小寫時區縮寫；僅適用於 TIMESTAMP TZ
OF	UTC 時差；僅適用於 TIMESTAMPTZ

> **Note**
>
> 您必須以單引號括住日期時間分隔符號 (例如 '-'、'/' 或 ':')，但必須以雙引號括住上表所列的「日期部分」和「時間部分」。

範例

如需將日期格式化為字串的範例，請參閱TO_CHAR。

數值格式字串

您可以在下面找到數字格式字串的參考。

下列格式字串適用於 TO_NUMBER 和 TO_CHAR 之類的函數。

- 如需將字串格式化為數字的範例，請參閱TO_NUMBER。
- 如需將數字格式化為字串的範例，請參閱TO_CHAR。

格式	描述
9	含指定位數的數值。
0	開頭為零的數值。
. (點)、D	小數點。
, (逗號)	千位分隔符號。
CC	世紀代碼。例如，第 21 世紀從 2001-01-01 開始 (僅適用於 TO_CHAR)。
FM	填補模式。禁止填補空格和零。
PR	角括號中的負值。
S	緊貼於數字的正負號。
L	指定位置中的貨幣符號。
G	群組分隔符號。
MI	在小於 0 的數字中，位於指定位置的減號。
PL	在大於 0 的數字中，位於指定位置的加號。
SG	指定位置中的加號或減號。
RN	介於 1 和 3999 之間的羅馬數字 (僅適用於 TO_CHAR)。
TH 或 th	序號字尾。不轉換小於零的小數或值。

數值資料的 Teradata 樣式格式化字元

接下來，您可以了解 TEXT_TO_INT_ALT 和 TEXT_TO_NUMERIC_ALT 函數如何解釋輸入運算式字串中的字元。您還可以找到可在 format 詞語中指定的字元清單。此外，您還可以找到 Teradata 樣式格式化和 Amazon Redshift 格式選項之間差異的說明。

格式	描述
G	輸入 expression 字串中不支援做為群組分隔符號。您無法在 format 詞語中指定此字元。
D	基數符號。您可以在 format 詞語中指定此字元。此字元相當於 .(句號)。 基數符號不能出現在包含下列任何字元的 format 詞語中： • .(句點) • S (大寫 's') • V (大寫 'v')
/ , : %	插入字元 / (正斜線)、逗號 (,)、: (冒號) 和 % (百分比符號)。 您不能在 format 詞語中包含這些字元。 Amazon Redshift 會忽略輸入 expression 字串中的這些字元。
.	句點作為基底數字元，即小數點。 此字元不能出現在以包含下列任何字元的 format 詞語中： • D (大寫 'd') • S (大寫 's') • V (大寫 'v')
B	您不能在 format 詞語中包含空格字元 (B)。在輸入 expression 字串中，開頭和結尾空格被忽略，並且不允許數字之間的空格。
+ -	您不能在 format 詞語中包含加號 (+) 或減號 (-)。但是，如果加號 (+) 和減號 (-) 出現在輸入

格式	描述
	expression 字串中，則會隱含地剖析為數值的一部分。
V	小數點位置指示器。 此字元不能出現在以包含下列任何字元的 format 詞語中： • D (大寫 'd') • .(句點)
Z	抑制零的十進位數字。Amazon Redshift 修剪開頭零。Z 字元不能跟在 9 字元後面。如果小數部分包含 9 字元，則 Z 字元必須位於基數字元的左側。
9	小數位。
CHAR(n)	對於此格式，您可以指定下列選項： • CHAR 由 Z 或 9 字元組成。Amazon Redshift 不支援 CHAR 值中的 + (加號) 或 - (減號)。 • n 是整數常數 I 或 F。對於 I，這是顯示數字或整數資料之整數部分所需的字元數。對於 F，這是顯示數值資料的小數部分所需的字元數。
-	連字號 (-) 字元。 您不能在 format 詞語中包含此字元。 Amazon Redshift 會忽略輸入 expression 字串中的這個字元。

格式	描述
S	帶符號的分區十進位值。S 字元必須位於 format 詞語中最後一個十進位數字之後。輸入 expression 字串的最後一個字元和對應的數字轉換在 用於有符號區十進位、Teradata 樣式數值資料格式的資料格式字元 中列出。 S 字元不能出現在包含下列任何字元的 format 詞語中： • + (加號) • .(句點) • D (大寫 'd') • Z (大寫 'z') • F (大寫 'f') • E (大寫 'e')
E	指數表示法。輸入 expression 字串可以包含指數字元。您不能在 format 詞語中指定 E 作為指數字元。
FN9	Amazon Redshift 不支援。
FNE	Amazon Redshift 不支援。
$、USD、美元	貨幣符號 ($)、ISO 貨幣符號 (USD)，以及貨幣名稱美元。 ISO 貨幣符號 USD 和貨幣名稱美元區分大小寫。Amazon Redshift 只支援美元貨幣。輸入 expression 字串可以包含美元貨幣符號與數值之間的空格，例如「$123E2」或「123E2 $」。
L	貨幣符號。此貨幣符號字元只能在 format 詞語中出現一次。您無法指定重複的貨幣符號字元。

格式	描述
C	ISO 貨幣符號。此貨幣符號字元只能在 format 詞語中出現一次。您無法指定重複的貨幣符號字元。
N	貨幣完整名稱。此貨幣符號字元只能在 format 詞語中出現一次。您無法指定重複的貨幣符號字元。
O	雙重貨幣符號。您無法在 format 詞語中指定此字元。
U	雙 ISO 貨幣符號。您無法在 format 詞語中指定此字元。
A	完整的雙重貨幣名稱。您無法在 format 詞語中指定此字元。

用於有符號區十進位、Teradata 樣式數值資料格式的資料格式字元

您可以在 TEXT_TO_INT_ALT 和 TEXT_TO_NUMERIC_ALT 函數的 format 詞語中使用以下字元來表示帶符號的分區十進位值。

輸入字串的最後一個字元	數值轉換
{ 或 0	n … 0
A 或 1	n … 1
B 或 2	n … 2
C 或 3	n … 3
D 或 4	n … 4
E 或 5	n … 5
F 或 6	n … 6

輸入字串的最後一個字元	數值轉換
G 或 7	n … 7
H 或 8	n … 8
I 或 9	n … 9
}	-n … 0
J	-n … 1
K	-n … 2
L	-n … 3
M	-n … 4
N	-n … 5
O	-n … 6
P	-n … 7
Q	-n … 8
R	-n … 9

日期和時間函數

在此節中，您可以找到 Amazon Redshift 支援的日期和時間 scalar 函數之相關資訊。

主題

- CONVERT_TIMEZONE 函數
- CURRENT_DATE 函數
- DATE_CMP 函數
- DATE_CMP_TIMESTAMP 函數
- DATE_CMP_TIMESTAMPTZ 函數
- DATEADD 函數
- DATEDIFF 函數
- DATE_PART 函數
- DATE_PART_YEAR 函數
- DATE_TRUNC 函數
- EXTRACT 函數
- GETDATE 函數
- INTERVAL_CMP 函數
- LAST_DAY 函數
- MONTHS_BETWEEN 函數
- NEXT_DAY 函數
- SYSDATE 函數
- TIMEOFDAY 函數
- TIMESTAMP_CMP 函數
- TIMESTAMP_CMP_DATE 函數
- TIMESTAMP_CMP_TIMESTAMPTZ 函數
- TIMESTAMPTZ_CMP 函數
- TIMESTAMPTZ_CMP_DATE 函數
- TIMESTAMPTZ_CMP_TIMESTAMP 函數
- TIMEZONE 函數
- TO_TIMESTAMP 函數
- TRUNC 函數
- 日期或時間戳記函數的日期部分

日期和時間函數的摘要

函式	語法	傳回值
+ (串連) 運算子 將日期與 + 符號兩側的時間串連起來，並傳回 TIMESTAMP 或 TIMESTAMPTZ。	date + time	TIMESTAMP 或 TIMESTAMPTZ
ADD_MONTHS 將指定幾個月新增至日期或時間戳記。	ADD_MONTHS ({date|timestamp}, integer)	TIMESTAMP
AT TIME ZONE 指定要透過 TIMESTAMP 或 TIMESTAMPTZ 表達式來使用哪一個時區。	AT TIME ZONE 'timezone'	TIMESTAMP 或 TIMESTAMPTZ
CONVERT_TIMEZONE 可將時間戳記從一個時區轉換為另一個時區。	CONVERT_TIMEZONE (['timezone',] 'timezone', timestamp)	TIMESTAMP
CURRENT_DATE 傳回目前工作階段時區中的日期 (預設為 UTC) 做為目前交易的開始。	CURRENT_DATE	DATE
DATE_CMP 比較兩個日期並傳回 0 (如果日期是相同的)、1 (如果 date1 較大)，且 -1 (如果 date2 較大)。	DATE_CMP (date1, date2)	INTEGER
DATE_CMP_TIMESTAMP 比較日期與時間並傳回 0 (如果值是相同的)、1 (如果 date 較大)，且 -1 (如果 timestamp 較大)。	DATE_CMP_TIMESTAMP (date, timestamp)	INTEGER
DATE_CMP_TIMESTAMPTZ	DATE_CMP_TIMESTAMPTZ (date, timestamptz)	INTEGER

函式	語法	傳回值
比較日期與含時區的時間戳記並傳回 0 (如果值是相同的)、1 (如果 date 較大)，且 -1 (如果 timestamptz 較大)。		
DATE_PART_YEAR 從日期擷取年份。	DATE_PART_YEAR (date)	INTEGER
DATEADD 透過指定間隔來增量日期或時間。	DATEADD (datepart, interval, {date|time|timetz|timestamp})	TIMESTAMP、TIME 或 TIMETZ
DATEDIFF 傳回兩個日期或時間的差異，做為給定日期的部分 (例如日或月)。	DATEDIFF (datepart, {date|time|timetz|timestamp}, {date|time|timetz|timestamp})	BIGINT
DATE_PART 從日期或時間擷取日期部分值。	DATE_PART (datepart, {date|timestamp})	DOUBLE
DATE_TRUNC 根據日期部分來截斷時間戳記。	DATE_TRUNC ('datepart', timestamp)	TIMESTAMP
EXTRACT 從 timestamp、timestamptz、time 或 timetz 中擷取日期或時間部分。	EXTRACT (datepart FROM source)	INTEGER or DOUBLE
GETDATE 傳回目前工作階段時區中的目前日期和時間 (預設為 UTC)。括號是必要的。	GETDATE()	TIMESTAMP

函式	語法	傳回值
INTERVAL_CMP 比較兩個間隔並傳回 0 (如果間隔是相同的)、1 (如果 interval1 較大)，且 -1 (如果 interval2 較大)。	INTERVAL_CMP (interval1, interval2)	INTEGER
LAST_DAY 傳回某月最後一天的日期，其中包含 date。	LAST_DAY(date)	DATE
MONTHS_BETWEEN 傳回兩個日期之間有幾個月。	MONTHS_BETWEEN (date, date)	FLOAT8
NEXT_DAY 傳回晚於 date 的第一個 day 執行個體的日期。	NEXT_DAY (date, day)	DATE
SYSDATE 傳回日期和時間 (以 UTC 表示) 做為目前交易的開始。	SYSDATE	TIMESTAMP
TIMEOFDAY 傳回目前工作階段時區中的目前工作日 (預設為 UTC) 做為字串值。	TIMEOFDAY()	VARCHAR
TIMESTAMP_CMP 比較兩個時間戳記並傳回 0 (如果時間戳記是相同的)、1 (如果 timestamp1 較大)，且 -1 (如果 timestamp2 較大)。	TIMESTAMP_CMP (timestamp1, timestamp2)	INTEGER
TIMESTAMP_CMP_DATE 比較時間戳記與日期並傳回 0 (如果值是相同的)、1 (如果 timestamp 較大)，且 -1 (如果 date 較大)。	TIMESTAMP_CMP_DATE (timestamp, date)	INTEGER

函式	語法	傳回值
TIMESTAMP_CMP_TIMESTAMPTZ 比較時間戳記與含時區的時間戳記並傳回 0 (如果值是相同的)、1 (如果 timestamp 較大)，且 -1 (如果 timestamptz 較大)。	TIMESTAMP_CMP_TIMESTAMPTZ (timestamp, timestamptz)	INTEGER
TIMESTAMPTZ_CMP 比較兩個含時區的時間戳記並傳回 0 (如果值是相同的)、1 (如果 timestamptz1 較大)，且 -1 (如果 timestamptz2 較大)。	TIMESTAMPTZ_CMP (timestamptz1, timestamptz2)	INTEGER
TIMESTAMPTZ_CMP_DATE 比較含時區的時間戳記值與日期並傳回 0 (如果值是相同的)、1 (如果 timestamptz 較大)，且 -1 (如果 date 較大)。	TIMESTAMPTZ_CMP_DATE (timestamptz, date)	INTEGER
TIMESTAMPTZ_CMP_TIMESTAMP 比較含時區的時間戳記與時間戳記並傳回 0 (如果值是相同的)、1 (如果 timestamptz 較大)，且 -1 (如果 timestamp 較大)。	TIMESTAMPTZ_CMP_TIMESTAMP (timestamptz, timestamp)	INTEGER
TIMEZONE 傳回時間戳記，做為指定時區和時間戳記值。	TIMEZONE ('timezone' { timestamp | timestamptz)	TIMESTAMP 或 TIMESTAMPTZ
TO_TIMESTAMP 傳回含時區的時間戳記，做為指定時間戳記和時區格式。	TO_TIMESTAMP ('timestamp', 'format')	TIMESTAMPTZ
TRUNC 截斷時間戳記並傳回日期。	TRUNC(timestamp)	DATE

> **Note**
>
> 不會將閏秒視為歷經時間的計算中。

交易中日期與時間函數

當您在交易區塊 (BEGIN … END) 中執行下列函數時，函數會傳回目前交易的開始時間，而不是目前陳述式的開始。

- SYSDATE
- TIMESTAMP
- CURRENT_DATE

下列函數一律會傳回目前陳述式的開始日期或時間 (即使他們在交易區塊中)。

- GETDATE
- TIMEOFDAY

已取代的僅限領導節點函數

下列日期函數已棄用，因為他們僅在領導節點上執行。如需詳細資訊，請參閱 僅限領導節點函數。

- AGE。請改用 DATEDIFF 函數。
- CURRENT_TIME。請改用 GETDATE 函數 或 SYSDATE。
- CURRENT_TIMESTAMP。請改用 GETDATE 函數 或 SYSDATE。
- LOCALTIME。請改用 GETDATE 函數 或 SYSDATE。
- LOCALTIMESTAMP。請改用 GETDATE 函數 或 SYSDATE。
- ISFINITE
- NOW。請改用 GETDATE 函數 或 SYSDATE。

+ (串連) 運算子

將 DATE 串連到 + 符號兩側的 TIME 或 TIMETZ，並傳回 TIMESTAMP 或 TIMESTAMPTZ。

語法

```
date + {time | timetz}
```

引數的順序可以反轉。例如，time + date。

引數

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。。

time

TIME 資料類型的欄，或是隱含評估為 TIME 類型的運算式。

timetz

TIMETZ 資料類型的欄，或是隱含評估為 TIMETZ 類型的運算式。

傳回類型

TIMESTAMP，如果輸入是 date + time。

TIMESTAMPTZ，如果輸入是 date + timetz。

範例

範例設定

若要設定範例中使用的 TIME_TEST 和 TIMTZ_TEST 資料表，請使用下列命令。

```
create table time_test(time_val time);

insert into time_test values
('20:00:00'),
('00:00:00.5550'),
('00:58:00');

create table timetz_test(timetz_val timetz);

insert into timetz_test values
('04:00:00+00'),
```

```
('00:00:00.5550+00'),
('05:58:00+00');
```

具有時間欄的範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME)，其中插入了三個值。

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

下列範例會串連日期常值和 TIME_VAL 欄。

```
select date '2000-01-02' + time_val as ts from time_test;

ts
---------------------
2000-01-02 20:00:00
2000-01-02 00:00:00.5550
2000-01-02 00:58:00
```

下列範例會串連日期常值和時間常值。

```
select date '2000-01-01' + time '20:00:00' as ts;

         ts
---------------------
 2000-01-01 20:00:00
```

下列範例會串連時間常值和日期常值。

```
select time '20:00:00' + date '2000-01-01' as ts;

         ts
---------------------
 2000-01-01 20:00:00
```

具有 TIMTZ 欄的範例

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ)，其中插入了三個值。

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會串連日期常值和 TIMETZ_VAL 欄。

```
select date '2000-01-01' + timetz_val as ts from timetz_test;
ts
---------------------
2000-01-01 04:00:00+00
2000-01-01 00:00:00.5550+00
2000-01-01 05:58:00+00
```

下列範例會串連 TIMETZ_VAL 欄和日期常值。

```
select timetz_val + date '2000-01-01' as ts from timetz_test;
ts
---------------------
2000-01-01 04:00:00+00
2000-01-01 00:00:00.5550+00
2000-01-01 05:58:00+00
```

下列範例會串連 DATE 常值和 TIMETZ 常值。此範例會傳回預設為 UTC 時區的 TIMESTAMPTZ。UTC 比 PST 早 8 小時，因此結果比輸入時間早 8 小時。

```
select date '2000-01-01' + timetz '20:00:00 PST' as ts;

          ts
------------------------
 2000-01-02 04:00:00+00
```

ADD_MONTHS 函數

ADD_MONTHS 會將指定幾個月新增至日期或時間戳記值或運算式。DATEADD 函數提供類似功能。

語法

```
ADD_MONTHS( {date | timestamp}, integer)
```

引數

date | timestamp

DATE 或 TIMESTAMP 資料類型的欄，或是隱含評估為 DATE 或 TIMESTAMP 類型的運算式。如果日期是某個月的最後一天，或者如果產生的月份是較短的月份，則函數會在結果中傳回該月的最後一天。若是其他日期，結果會包含與日期表達式一樣的相同天數。

integer

INTEGER 資料類型的值。使用負數可減少日期中的月份。

傳回類型

TIMESTAMP

範例

以下查詢會使用 TRUNC 函數中的 ADD_MONTHS 函數。TRUNC 函數會從 ADD_MONTHS 結果移除某日時間。ADD_MONTHS 函數會從 CALDATE 欄中新增 12 個月至每個值。在 CALDATE 欄中的值是日期。

```
select distinct trunc(add_months(caldate, 12)) as calplus12,
trunc(caldate) as cal
from date
order by 1 asc;

 calplus12  |    cal
------------+------------
 2009-01-01 | 2008-01-01
 2009-01-02 | 2008-01-02
 2009-01-03 | 2008-01-03
...
(365 rows)
```

下列範例會使用 ADD_MONTHS 函數，將 1 個月加到 timestamp。

```
select add_months('2008-01-01 05:07:30', 1);
```

```
add_months
---------------------
2008-02-01 05:07:30
```

下列範例示範當 ADD_MONTHS 函數對含有月份的日期進行操作的行為，而這些月份的天數皆不同時。此範例顯示函數如何處理在 3 月 31 日增加 1 個月，以及在 4 月 30 日增加 1 個月。4 月有 30 天，因此在 3 月 31 日的基礎上加上 1 個月就得到 4 月 30 日。5 月有 31 天，因此在 4 月 30 日的基礎上加上 1 個月就得到 5 月 31 日。

```
select add_months('2008-03-31',1);

add_months
---------------------
2008-04-30 00:00:00

select add_months('2008-04-30',1);

add_months
---------------------
2008-05-31 00:00:00
```

AT TIME ZONE 函數

AT TIME ZONE 指定要透過 TIMESTAMP 或 TIMESTAMPTZ 表達式來使用哪一個時區。

語法

```
AT TIME ZONE 'timezone'
```

引數

timezone

傳回值的 TIMEZONE。您可以將時區指定為時區名稱 (例如 **'Africa/Kampala'** 或 **'Singapore'**) 或做為時區縮寫 (例如 **'UTC'** 或 **'PDT'**)。

若要查看受支援時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```

若要查看受支援時區縮寫的清單，請執行下列命令。

```
select pg_timezone_abbrevs();
```

如需詳細資訊和範例，請參閱 時區使用須知。

傳回類型

與 TIMESTAMP 表達式搭配使用時的 TIMESTAMPTZ。與 TIMESTAMPTZ 表達式搭配使用時的 TIMESTAMP。

範例

下列範例會轉換不含時區的時間戳記值，並將其解譯為 MST 時間 (POSIX 中的 UTC+7)。此範例會傳回 UTC 時區的 TIMESTAMPTZ 資料類型值。如果您將預設時區設定為 UTC 以外的時區，可能會看到不同的結果。

```
SELECT TIMESTAMP '2001-02-16 20:38:40' AT TIME ZONE 'MST';

timezone
------------------------
2001-02-17 03:38:40+00
```

下列範例會使用含時區值的輸入時間戳記，其中指定的時區是 EST (POSIX 中的 UTC+5) 且會將其轉換為 MST (POSIX 中的 UTC+7)。此範例會傳回 TIMESTAMP 資料類型的值。

```
SELECT TIMESTAMPTZ '2001-02-16 20:38:40-05' AT TIME ZONE 'MST';

timezone
------------------------
2001-02-16 18:38:40
```

CONVERT_TIMEZONE 函數

CONVERT_TIMEZONE 可將時間戳記從一個時區轉換為另一個時區。此功能會根據日光節約時間自動調整。

語法

```
CONVERT_TIMEZONE( ['source_timezone',] 'target_timezone', 'timestamp')
```

引數

source_timezone

(選用) 目前時間戳記的時區。預設值為 UTC。如需詳細資訊，請參閱 時區使用須知。

target_timezone

新時間戳記的時區。如需詳細資訊，請參閱 時區使用須知。

timestamp

時間戳記欄或運算式會隱性轉換為時間戳記。

傳回類型

TIMESTAMP

時區使用須知

來源時區或 target_ 時區/時區可以指定為時區名稱 (例如「非洲/坎帕拉」或「新加坡」) ，也可以指定為時區縮寫 (例如「世界標準時間」或「PDT」) 。您不必將時區名稱轉換為名稱或縮寫轉換為縮寫。例如，您可以從來源時區名稱「Singapore」中選擇時間戳記，並將其轉換為時區縮寫「PDT」的時間戳記。

> **Note**
>
> 使用時區名稱或時區縮寫的結果可能會因當地季節時間 (例如夏令時間) 而有所不同。

使用時區名稱

若要檢視目前和完整的時區名稱清單，請執行下列命令。

```
select pg_timezone_names();
```

每一個資料列都包含一個逗號分隔的字串，其中包含時區名稱、縮寫、UTC 時差，以及時區是否會遵循日光節約時間 (t 或 f) 的指標。以下列程式碼片段為例，其將顯示兩個產生的資料列。第一行是時區Europe/Paris，縮寫CET，與 UTC 的01:00:00偏移量，並f表示它不會觀察日光節約時間。第二行是時區Israel，縮寫IST，與 UTC 02:00:00 偏移量，並f表示它不會觀察日光節約時間。

```
pg_timezone_names
-----------------
(Europe/Paris,CET,01:00:00,f)
(Israel,IST,02:00:00,f)
```

執行 SQL 陳述式以取得整個清單並尋找時區名稱。大約會傳回 600 個資料列。雖然部分傳回的時區名稱為大寫首字母或縮寫 (例如，GB、PRC、ROK)，CONVERT_TIMEZONE 功能仍會將其視為時區名稱，而非時區縮寫。

如果您使用時區名稱指定時區，CONVERT_TIMZONE 會自動調整夏令時 (DST) 或任何其他本地季節性協定，例如夏令時間、標準時間或冬令時間，這些時區在「timestamp」指定的日期和時間內生效。例如，'Europe/London' 在冬天代表 UTC，在夏天增加一小時。

使用時區縮寫

若要檢視目前和完整的時區縮寫清單，請執行下列命令。

```
select pg_timezone_abbrevs();
```

結果都會包含一個逗號分隔的字串，其中包含時區縮寫、UTC 時差，以及時區是否會遵循日光節約時間的指標 (t 或 f)。以下列程式碼片段為例，其將顯示兩個產生的資料列。第一列包含太平洋夏令時間的縮寫 (PDT)，與 UTC 的時差為 -07:00:00，以及表示其遵循日光節約時間的 t。第二列包含太平洋標準時間的縮寫PST，與 UTC 的-08:00:00偏移量，並表示它不會觀察f到節省日光的時間。

```
pg_timezone_abbrevs
--------------------
(PDT,-07:00:00,t)
(PST,-08:00:00,f)
```

執行 SQL 陳述式以取得整個清單，並根據其時差和日光節約時間指標來尋找縮寫。大約會傳回 200 個資料列。

時區縮寫代表與 UTC 的固定偏差。如果您使用時區縮寫指定時區，CONVERT_TIMZONE 會使用與 UTC 相比的固定偏移量，並且不會針對任何本地季節性通訊協定進行調整。

使用 POSIX 樣式格式

POSIX 樣式時區規格的格式是 STDoffset 或 STDoffsetDST，其中 STD 是時區縮寫，offset 是 UTC 以西的小時偏移量，DST 是選用的日光節約時區縮寫。日光節約時間假設為較給定偏移早一小時。

POSIX 樣式時區格式使用格林威治以西的正偏移，與 ISO-8601 慣例不同，此是使用格林威治以東的正偏移。

以下是 POSIX 樣式時區的範例：

- PST8
- PST8PDT
- EST5
- EST5EDT

> **Note**
>
> Amazon Redshift 不會驗證 POSIX 樣式時區規格，所以可能會將時區設為無效值。例如，即使將時區設為無效值，下列命令不會傳回錯誤。
>
> ```
> set timezone to ‘xxx36’;
> ```

範例

許多範例都使用 TICKIT 範例資料集。如需詳細資訊， 請參閱範本資料庫。

以下範例會將時間戳記值從預設 UTC 時區轉換為 PST。

```
select convert_timezone('PST', '2008-08-21 07:23:54');

 convert_timezone
-------------------------
2008-08-20 23:23:54
```

以下範例會將 LISTTIME 欄位中的時間戳記值從預設 UTC 時區轉換為 PST。即使時間戳記是在日光節約時間期間，其會轉換為標準時間，因為目標時區是指定為縮寫 (PST)。

```
select listtime, convert_timezone('PST', listtime) from listing
where listid = 16;

     listtime       |   convert_timezone
--------------------+-------------------
```

```
2008-08-24 09:36:12     2008-08-24 01:36:12
```

以下範例會將時間戳記 LISTTIME 欄位從預設 UTC 時區轉換為 US/Pacific 時區。目標時區使用時區名稱，且時間戳記是在日光節約時間期間，因此函數會傳回日光時間。

```
select listtime, convert_timezone('US/Pacific', listtime) from listing
where listid = 16;

     listtime       |   convert_timezone
--------------------+---------------------
2008-08-24 09:36:12 | 2008-08-24 02:36:12
```

以下範例會將時間戳記字串從 EST 轉換為 PST：

```
select convert_timezone('EST', 'PST', '20080305 12:25:29');

 convert_timezone
-------------------
2008-03-05 09:25:29
```

以下範例會將時間戳記轉換為美國東部標準時間，因為目標時區使用時區名稱 (America/New_York) 且時間戳記是在標準時間期間。

```
select convert_timezone('America/New_York', '2013-02-01 08:00:00');

 convert_timezone
---------------------
2013-02-01 03:00:00
(1 row)
```

以下範例會將時間戳記轉換為美國東部日光時間，因為目標時區使用時區名稱 (America/New_York) 且時間戳記是在日光時間期間。

```
select convert_timezone('America/New_York', '2013-06-01 08:00:00');

 convert_timezone
---------------------
2013-06-01 04:00:00
(1 row)
```

以下範例示範的是偏移的使用。

```
SELECT CONVERT_TIMEZONE('GMT','NEWZONE +2','2014-05-17 12:00:00') as newzone_plus_2,
CONVERT_TIMEZONE('GMT','NEWZONE-2:15','2014-05-17 12:00:00') as newzone_minus_2_15,
CONVERT_TIMEZONE('GMT','America/Los_Angeles+2','2014-05-17 12:00:00') as la_plus_2,
CONVERT_TIMEZONE('GMT','GMT+2','2014-05-17 12:00:00') as gmt_plus_2;

   newzone_plus_2    | newzone_minus_2_15  |      la_plus_2      |     gmt_plus_2
---------------------+---------------------+---------------------+---------------------
2014-05-17 10:00:00 | 2014-05-17 14:15:00 | 2014-05-17 10:00:00 | 2014-05-17 10:00:00
(1 row)
```

CURRENT_DATE 函數

CURRENT_DATE 傳回目前工作階段時區中的日期 (預設為 UTC)，使用預設格式：YYYY-MM-DD。

> **Note**
>
> CURRENT_DATE 傳回目前交易的日期 (而不是目前陳述式的開始)。考慮以下場景：您在 10/01/08 23:59 啟動包含多個陳述式的交易，而包含 CURRENT_DATE 的陳述式在 10/02/08 00:00 執行。CURRENT_DATE 傳回 10/01/08，而不是 10/02/08。

語法

```
CURRENT_DATE
```

傳回類型

DATE

範例

下列範例會傳回目前的日期 (函式執行的 AWS 區域 位置)。

```
select current_date;

   date
------------
2008-10-01
```

下列範例會建立資料表，插入 todays_date 欄預設值為 CURRENT_DATE 的列，然後選取資料表中的所有列。

```
CREATE TABLE insert_dates(
    label varchar(128) NOT NULL,
    todays_date DATE DEFAULT CURRENT_DATE);

INSERT INTO insert_dates(label)
VALUES('Date row inserted');

SELECT * FROM insert_dates;


 label            | todays_date
------------------+-------------
Date row inserted | 2023-05-10
```

DATE_CMP 函數

DATE_CMP 會比較兩個日期。該函數會傳回 0 (如果日期是相同的)、1 (如果 date1 較大)，且 -1 (如果 date2 較大)。

語法

```
DATE_CMP(date1, date2)
```

引數

date1

DATE 資料類型的欄，或是評估為 DATE 類型的運算式。

date2

DATE 資料類型的欄，或是評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

下列查詢會比較 CALDATE 欄位的 DATE 值與日期 2008 年 1 月 4 日，並傳回 CALDATE 中的值是早於 (-1)、等於 (0) 或晚於 (1) 2008 年 1 月 4 日：

```
select caldate, '2008-01-04',
date_cmp(caldate,'2008-01-04')
from date
order by dateid
limit 10;

 caldate   |  ?column?  | date_cmp
-----------+------------+----------
2008-01-01 | 2008-01-04 |       -1
2008-01-02 | 2008-01-04 |       -1
2008-01-03 | 2008-01-04 |       -1
2008-01-04 | 2008-01-04 |        0
2008-01-05 | 2008-01-04 |        1
2008-01-06 | 2008-01-04 |        1
2008-01-07 | 2008-01-04 |        1
2008-01-08 | 2008-01-04 |        1
2008-01-09 | 2008-01-04 |        1
2008-01-10 | 2008-01-04 |        1
(10 rows)
```

DATE_CMP_TIMESTAMP 函數

DATE_CMP_TIMESTAMP 會比較日期與時間戳記並傳回 0 (如果值是相同的)、1 (如果 date 較大)，以及 -1 (如果 timestamp 較大)。

語法

```
DATE_CMP_TIMESTAMP(date, timestamp)
```

引數

date

DATE 資料類型的欄，或是評估為 DATE 類型的運算式。

timestamp

TIMESTAMP 資料類型的欄，或是評估為 TIMESTAMP 類型的運算式。

傳回類型

INTEGER

範例

例如，下列範例會比較日期 2008-06-18 與 LISTTIME。LISTTIME 欄的值是時間戳記。在此日期前所做的清單會傳回 1，在此日期後所做的清單會傳回 -1。

```
select listid, '2008-06-18', listtime,
date_cmp_timestamp('2008-06-18', listtime)
from listing
order by 1, 2, 3, 4
limit 10;

 listid |  ?column?  |      listtime       | date_cmp_timestamp
--------+------------+---------------------+--------------------
      1 | 2008-06-18 | 2008-01-24 06:43:29 |                  1
      2 | 2008-06-18 | 2008-03-05 12:25:29 |                  1
      3 | 2008-06-18 | 2008-11-01 07:35:33 |                 -1
      4 | 2008-06-18 | 2008-05-24 01:18:37 |                  1
      5 | 2008-06-18 | 2008-05-17 02:29:11 |                  1
      6 | 2008-06-18 | 2008-08-15 02:08:13 |                 -1
      7 | 2008-06-18 | 2008-11-15 09:38:15 |                 -1
      8 | 2008-06-18 | 2008-11-09 05:07:30 |                 -1
      9 | 2008-06-18 | 2008-09-09 08:03:36 |                 -1
     10 | 2008-06-18 | 2008-06-17 09:44:54 |                  1
(10 rows)
```

DATE_CMP_TIMESTAMPTZ 函數

DATE_CMP_TIMESTAMPTZ 會比較日期與含時區的時間戳記，並傳回 0 (如果值是相同的)、1 (如果 date 較大)，以及 -1 (如果 timestamptz 較大)。

語法

```
DATE_CMP_TIMESTAMPTZ(date, timestamptz)
```

引數

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。。

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

傳回類型

INTEGER

範例

例如，下列範例會比較日期 2008-06-18 與 LISTTIME。在此日期前所做的清單會傳回 1，在此日期後所做的清單會傳回 -1。

```
select listid, '2008-06-18', CAST(listtime AS timestamptz),
date_cmp_timestamptz('2008-06-18', CAST(listtime AS timestamptz))
from listing
order by 1, 2, 3, 4
limit 10;

 listid |  ?column?  |      timestamptz       | date_cmp_timestamptz
--------+------------+------------------------+----------------------
      1 | 2008-06-18 | 2008-01-24 06:43:29+00 |                    1
      2 | 2008-06-18 | 2008-03-05 12:25:29+00 |                    1
      3 | 2008-06-18 | 2008-11-01 07:35:33+00 |                   -1
      4 | 2008-06-18 | 2008-05-24 01:18:37+00 |                    1
      5 | 2008-06-18 | 2008-05-17 02:29:11+00 |                    1
      6 | 2008-06-18 | 2008-08-15 02:08:13+00 |                   -1
      7 | 2008-06-18 | 2008-11-15 09:38:15+00 |                   -1
      8 | 2008-06-18 | 2008-11-09 05:07:30+00 |                   -1
      9 | 2008-06-18 | 2008-09-09 08:03:36+00 |                   -1
     10 | 2008-06-18 | 2008-06-17 09:44:54+00 |                    1
(10 rows)
```

DATEADD 函數

透過指定間隔來增量 DATE、TIME、TIMETZ 或 TIMESTAMP 值。

語法

```
DATEADD( datepart, interval, {date|time|timetz|timestamp} )
```

引數

datepart

函數對其執行的日期部分 (例如，年、月、日或小時)。如需詳細資訊，請參閱 日期或時間戳記函數的日期部分。

間隔

指定要新增至目標表達式之間隔 (例如，天數) 的整數。負整數會減去間隔。

date|time|timetz|timestamp

DATE、TIME、TIMETZ 或 TIMESTAMP 欄或隱含轉換為 DATE、TIME、TIMETZ 或 TIMESTAMP 的運算式。DATE、TIME、TIMETZ 或 TIMESTAMP 運算式必須包含指定的日期部分。

傳回類型

TIMESTAMP 或 TIME 或 TIMETZ 取決於輸入資料類型。

具有 DATE 欄的範例

下列範例會向 DATE 資料表中存在的 11 月份的每個日期增加 30 天。

```
select dateadd(day,30,caldate) as novplus30
from date
where month='NOV'
order by dateid;

novplus30
---------------------
2008-12-01 00:00:00
2008-12-02 00:00:00
2008-12-03 00:00:00
...
(30 rows)
```

下列範例會將 18 個月增加至常值日期值。

```
select dateadd(month,18,'2008-02-28');

date_add
---------------------
2009-08-28 00:00:00
(1 row)
```

DATEADD 函數的預設欄名稱為 DATE_ADD。日期值的預設時間戳記為 00:00:00。

下列範例會將 30 分鐘增加至未指定時間戳記的日期值。

```
select dateadd(m,30,'2008-02-28');

date_add
---------------------
2008-02-28 00:30:00
(1 row)
```

您可以用全名或縮寫來表示日期部分。在這種情況下，m 代表分鐘，而不是月。

具有 TIME 欄的範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME)，其中插入了三個值。

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

下列範例會將 5 分鐘增加至 TIME_TEST 資料表中的每個 TIME_VAL。

```
select dateadd(minute,5,time_val) as minplus5 from time_test;

minplus5
---------------
20:05:00
00:05:00.5550
01:03:00
```

下列範例會將 8 小時增加至常值時間值。

```
select dateadd(hour, 8, time '13:24:55');

date_add
---------------
21:24:55
```

下列範例會顯示時間超過 24:00:00 或在 00:00:00 以下的時間。

```
select dateadd(hour, 12, time '13:24:55');

date_add
---------------
01:24:55
```

具有 TIMTZ 欄的範例

這些範例中的輸出值是以 UTC 為預設時區。

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ)，其中插入了三個值。

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會將 5 分鐘增加至 TIMETZ_TEST 資料表中的每個 TIMETZ_VAL。

```
select dateadd(minute,5,timetz_val) as minplus5_tz from timetz_test;

minplus5_tz
---------------
04:05:00+00
00:05:00.5550+00
06:03:00+00
```

下列範例會將 2 小時加入常值 timetz 值。

```
select dateadd(hour, 2, timetz '13:24:55 PST');

date_add
---------------
23:24:55+00
```

具有 TIMESTAMP 欄的範例

這些範例中的輸出值是以 UTC 為預設時區。

下列範例資料表 TIMESTAMP_TEST 有一個 TIMESTAMP_VAL 欄 (類型為 TIMESTAMP)，其中插入了三個值。

```
SELECT timestamp_val FROM timestamp_test;

timestamp_val
------------------
1988-05-15 10:23:31
2021-03-18 17:20:41
2023-06-02 18:11:12
```

以下範例僅將 20 年增加至 2000 年之前的 TIMESTAMP_TEST 中的 TIMESTAMP_VAL 值。

```
SELECT dateadd(year,20,timestamp_val)
FROM timestamp_test
WHERE timestamp_val < to_timestamp('2000-01-01 00:00:00', 'YYYY-MM-DD HH:MI:SS');

date_add
---------------
2008-05-15 10:23:31
```

下列範例將 5 秒加入不帶秒指示器的常值時間戳記值。

```
SELECT dateadd(second, 5, timestamp '2001-06-06');

date_add
---------------
2001-06-06 00:00:05
```

使用須知

DATEADD(month, ...) 和 ADD_MONTHS 函數處理落在月底之日期的方式會所有不同：

- ADD_MONTHS：如果您要新增的日期是某個月的最後一天，則產生的月份總是產生月份的最後一天，不論該月份的長短。例如，4 月 30 號 + 1 個月是 5 月 31 號。

  ```
  select add_months('2008-04-30',1);

  add_months
  ---------------------
  2008-05-31 00:00:00
  ```

```
(1 row)
```

- DATEADD：如果您要新增之日期中的天數少於結果月份，則結果將會是結果月份的相對應天數，而不會是該月的最後一天。例如，4 月 30 號 + 1 個月是 5 月 30 號。

```
select dateadd(month,1,'2008-04-30');

date_add
---------------------
2008-05-30 00:00:00
(1 row)
```

DATEADD 函數使用 dateadd (月份，12...) 或 dateadd (年份，1...) 時，處理閏年日期 02-29 的方式會不同。

```
select dateadd(month,12,'2016-02-29');

date_add
---------------------
2017-02-28 00:00:00

select dateadd(year, 1, '2016-02-29');

date_add
---------------------
2017-03-01 00:00:00
```

DATEDIFF 函數

DATEDIFF 傳回兩個日期或時間表達式之日期部分的差異。

語法

```
DATEDIFF( datepart, {date|time|timetz|timestamp}, {date|time|timetz|timestamp} )
```

引數

datepart

函數運作的日期或時間值 (年、月或日、小時、分鐘、秒、毫秒或微秒) 的特定部分。如需詳細資訊，請參閱 日期或時間戳記函數的日期部分。

特別是，DATEDIFF 會決定兩個運算式間相距的日期部分邊界數。例如，假設您正在計算兩個日期 12-31-2008 和 01-01-2009 之間的年度差異。在這種情況下，函數傳回 1 年，儘管這些日期只相隔一天。如果您正在尋找兩個時間戳記間時數的差異，01-01-2009 8:30:00 和 01-01-2009 10:00:00，結果為 2 小時。如果您正在尋找兩個時間戳記間時數的差異，8:30:00 和 10:00:00，結果為 2 小時。

date|time|timetz|timestamp

DATE、TIME、TIMETZ 或 TIMESTAMP 欄，或隱含轉換為 DATE、TIME、TIMETZ 或 TIMESTAMP 的運算式。運算式必須同時包含指定的日期部分或時間部分。如果第二個日期或時間晚於第一個日期或時間，結果為正值。如果第二個日期或時間早於第一個日期或時間，結果為負值。

傳回類型

BIGINT

具有 DATE 欄的範例

下列範例會尋找在兩個常值日期值間週次的差異。

```
select datediff(week,'2009-01-01','2009-12-31') as numweeks;

numweeks
----------
52
(1 row)
```

下列範例會尋找兩個常值日期值之間的差異 (以小時為單位)。如果您未提供日期的時間值，則預設值為 00:00:00。

```
select datediff(hour, '2023-01-01', '2023-01-03 05:04:03');

date_diff
----------
53
(1 row)
```

下列範例會尋找兩個常值 TIMESTAMETZ 值之間的差異 (以天為單位)。

```
Select datediff(days, 'Jun 1,2008  09:59:59 EST', 'Jul 4,2008  09:59:59 EST')
```

```
date_diff
----------
33
```

以下範例找出資料表中同一列兩個日期之間的差異 (以天為單位)。

```
select * from date_table;

start_date |   end_date
-----------+-----------
2009-01-01 | 2009-03-23
2023-01-04 | 2024-05-04
(2 rows)

select datediff(day, start_date, end_date) as duration from date_table;

duration
---------
      81
     486
(2 rows)
```

下列範例會尋找在過去和今日日期中常值間季次的差異。此範例假設目前日期為 2008 年 6 月 5 日。您可以用全名或縮寫來表示日期部分。DATEDIFF 函數的預設欄名稱為 DATE_DIFF。

```
select datediff(qtr, '1998-07-01', current_date);

date_diff
-----------
40
(1 row)
```

下列範例會聯結 SALES 和 LISTING 資歷表，來計算在他們列出和門票售出後所經的天數：清單 1000 到 1005。這些清單銷售的最長等待時間是 15 天，而最短時間是不到一天 (0 天)。

```
select priceperticket,
datediff(day, listtime, saletime) as wait
from sales, listing where sales.listid = listing.listid
and sales.listid between 1000 and 1005
order by wait desc, priceperticket desc;
```

```
priceperticket | wait
---------------+------
 96.00         |   15
 123.00        |   11
 131.00        |    9
 123.00        |    6
 129.00        |    4
 96.00         |    4
 96.00         |    0
(7 rows)
```

此範例會計算賣家等待所有門票特價的平均小時數。

```
select avg(datediff(hours, listtime, saletime)) as avgwait
from sales, listing
where sales.listid = listing.listid;

avgwait
---------
465
(1 row)
```

具有 TIME 欄的範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME)，其中插入了三個值。

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
00:58:00
```

下列範例會尋找 TIME_VAL 欄與時間常值之間的小時數差異。

```
select datediff(hour, time_val, time '15:24:45') from time_test;

 date_diff
-----------
        -5
```

```
        15
        15
```

下列範例會尋找兩個常值時間值之間的分鐘數差異。

```
select datediff(minute, time '20:00:00', time '21:00:00') as nummins;

nummins
----------
60
```

具有 TIMTZ 欄的範例

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ)，其中插入了三個值。

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會尋找 TIMETZ 常值與 timetz_val 之間的小時數差異。

```
select datediff(hours, timetz '20:00:00 PST', timetz_val) as numhours from timetz_test;

numhours
----------
0
-4
1
```

下列範例會尋找兩個常值 TIMTZ 值之間的小時數差異。

```
select datediff(hours, timetz '20:00:00 PST', timetz '00:58:00 EST') as numhours;

numhours
----------
1
```

DATE_PART 函數

DATE_PART 會從運算式擷取日期部分值。DATE_PART 是 PGDATE_PART 函數的同義詞。

語法

```
DATE_PART(datepart, {date|timestamp})
```

引數

datepart

函數所操作的日期值的特定部分 (例如年、月或日) 的識別碼常值或字串。如需詳細資訊，請參閱 日期或時間戳記函數的日期部分。

{date|timestamp}

日期欄、時間戳記欄，或會隱性轉換為日期或時間戳記的運算式。Date 或 timestamp 中的欄或運算式必須包含 datepart 中指定的日期部分。

傳回類型

DOUBLE

範例

DATE_PART 函數的預設欄名稱為 pgdate_part。

如需其中一些範例中所使用資料的相關資訊，請參閱 範本資料庫。

下列範例會從時間戳記常值尋找分鐘。

```
SELECT DATE_PART(minute, timestamp '20230104 04:05:06.789');

pgdate_part
-----------
          5
```

下列範例會從時間戳記常值中尋找週數。週數計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 ISO 8601。

```
SELECT DATE_PART(week, timestamp '20220502 04:05:06.789');
```

```
pgdate_part
-----------
         18
```

下列範例會從時間戳記常值尋找月份中的日期。

```
SELECT DATE_PART(day, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
         2
```

下列範例會從時間戳記常值尋找星期幾。週數計算是從 0-6 開始的整數，以星期日開始。

```
SELECT DATE_PART(dayofweek, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
         1
```

下列範例會從時間戳記常值中尋找世紀。世紀計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 ISO 8601。

```
SELECT DATE_PART(century, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
         21
```

下列範例會從時間戳記常值中尋找千年。千年計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 ISO 8601。

```
SELECT DATE_PART(millennium, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
         3
```

下列範例會從時間戳記常值中尋找微秒。微秒計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 ISO 8601。

```
SELECT DATE_PART(microsecond, timestamp '20220502 04:05:06.789');

pgdate_part
-----------
    789000
```

下列範例會從日期常值中尋找月份。

```
SELECT DATE_PART(month, date '20220502');

pgdate_part
-----------
          5
```

下列範例會將 DATE_PART 函數套用至資料表中的欄。

```
SELECT date_part(w, listtime) AS weeks, listtime
FROM listing
WHERE listid=10


weeks |      listtime
------+---------------------
 25   | 2008-06-17 09:44:54
(1 row)
```

您可以將日期部分以全名或縮寫表示，在此情況下，w 代表週。

週日期部分的天會傳回從 0–6 的整數，從星期日開始。使用 DATE_PART 含小數點 (DAYOFWEEK)，來檢視星期六的事件。

```
SELECT date_part(dow, starttime) AS dow, starttime
FROM event
WHERE date_part(dow, starttime)=6
ORDER BY 2,1;

 dow |      starttime
-----+---------------------
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
   6 | 2008-01-05 14:00:00
```

```
   6 | 2008-01-05 14:00:00
...
(1147 rows)
```

DATE_PART_YEAR 函數

DATE_PART_YEAR 函數從日期擷取年份。

語法

```
DATE_PART_YEAR(date)
```

引數

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

下列範例會從日期常值中尋找年份。

```
SELECT DATE_PART_YEAR(date '20220502 04:05:06.789');

date_part_year
---------------
2022
```

下列範例從 CALDATE 欄位中擷取年份。在 CALDATE 欄中的值是日期。如需此範例中所使用資料的相關資訊，請參閱 範本資料庫。

```
select caldate, date_part_year(caldate)
from date
order by
dateid limit 10;
```

```
 caldate   | date_part_year
-----------+----------------
2008-01-01 |           2008
2008-01-02 |           2008
2008-01-03 |           2008
2008-01-04 |           2008
2008-01-05 |           2008
2008-01-06 |           2008
2008-01-07 |           2008
2008-01-08 |           2008
2008-01-09 |           2008
2008-01-10 |           2008
(10 rows)
```

DATE_TRUNC 函數

DATE_TRUNC 函數會根據您指定的日期部分 (例如小時、天或月) 來截斷時間戳記運算式或常值。

語法

```
DATE_TRUNC('datepart', timestamp)
```

引數

datepart

時間戳記值截斷目標的日期部分。輸入 timestamp 被截斷為輸入 datepart 的精確度。例如，month 截斷為每個月的第一天。有效格式如下：

- microsecond, microseconds
- millisecond, milliseconds
- second, seconds
- minute, minutes
- hour, hours
- day, days
- week, weeks
- month, months
- quarter, quarters
- year, years

- decade, decades
- century, centuries
- millennium, millennia

如需某些格式縮寫的相關資訊，請參閱 日期或時間戳記函數的日期部分

timestamp

時間戳記欄或運算式會隱性轉換為時間戳記。

傳回類型

TIMESTAMP

範例

將輸入時間戳記截斷為秒。

```
SELECT DATE_TRUNC('second', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:06
```

將輸入時間戳記截斷為分鐘。

```
SELECT DATE_TRUNC('minute', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:00
```

將輸入時間戳記截斷為小時。

```
SELECT DATE_TRUNC('hour', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:00:00
```

將輸入時間戳記截斷為天。

```
SELECT DATE_TRUNC('day', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 00:00:00
```

將輸入時間戳記截斷為一個月的第一天。

```
SELECT DATE_TRUNC('month', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-01 00:00:00
```

將輸入時間戳記截斷為季度的第一天。

```
SELECT DATE_TRUNC('quarter', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-01 00:00:00
```

將輸入時間戳記截斷為一年的第一天。

```
SELECT DATE_TRUNC('year', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-01-01 00:00:00
```

將輸入時間戳記截斷為世紀的第一天。

```
SELECT DATE_TRUNC('millennium', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2001-01-01 00:00:00
```

將輸入時間戳記截斷為一週的星期一。

```
select date_trunc('week', TIMESTAMP '20220430 04:05:06.789');
date_trunc
2022-04-25 00:00:00
```

在下列範例中，DATE_TRUNC 函數會使用「週」日期部分來傳回每週星期一的日期。

```
select date_trunc('week', saletime), sum(pricepaid) from sales where
saletime like '2008-09%' group by date_trunc('week', saletime) order by 1;

date_trunc  |    sum
------------+-------------
2008-09-01  | 2474899
2008-09-08  | 2412354
2008-09-15  | 2364707
```

```
2008-09-22  | 2359351
2008-09-29  |  705249
```

EXTRACT 函數

該 EXTRACT 函數從時間戳，TIMSTAMPTZ，時間，時間，時間，間隔年到月，或間隔天到第二個值返回日期或時間部分。範例包括時間戳記中的日、月、年、時、分、秒、毫秒或微秒。

語法

```
EXTRACT(datepart FROM source)
```

引數

datepart

要擷取的日期或時間的分欄，例如日、月、年、小時、分鐘、秒、毫秒或微秒。對於可能的值，請參閱 日期或時間戳記函數的日期部分。

source

評估為「時間戳記」、「TIMESTAMPTZ」、「時間」、「時間」、「時間」、「年到月的間隔」或「從日到秒的間隔」資料類型的資料欄或運算式。

傳回類型

INTEGER 如果來源值評估為資料類型「時間戳記」、「時間」、「時間」、「年到月的間隔」或「從日到秒的間隔」。

如果 source 值計算為資料類型 TIMESTAMPTZ，則為 DOUBLE PRECISION。

TIMESTAMP 範例

下列範例會判斷特價時，售價是 10,000 USD 或更多的週次。此範例使用 TICKIT 資料。如需詳細資訊，請參閱 範本資料庫。

```
select salesid, extract(week from saletime) as weeknum
from sales
where pricepaid > 9999
order by 2;

salesid | weeknum
```

```
--------+---------
 159073 |       6
 160318 |       8
 161723 |      26
```

下列範例會從常值 timestamp 值傳回分鐘值。

```
select extract(minute from timestamp '2009-09-09 12:08:43');

date_part
-----------
8
```

下列範例會從常值 timestamp 值傳回毫秒值。

```
select extract(ms from timestamp '2009-09-09 12:08:43.101');

date_part
-----------
101
```

TIMESTAMPTZ 範例

下列範例會從常值 timestamptz 傳回年份值。

```
select extract(year from timestamptz '1.12.1997 07:37:16.00 PST');

date_part
-----------
1997
```

TIME 範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME)，其中插入了三個值。

```
select time_val from time_test;

time_val
---------------------
20:00:00
00:00:00.5550
```

```
00:58:00
```

下列範例會擷取每個 time_val 的分鐘。

```
select extract(minute from time_val) as minutes from time_test;

minutes
-----------
         0
         0
         58
```

下列範例會擷取每個 time_val 中的小時數。

```
select extract(hour from time_val) as hours from time_test;

hours
-----------
         20
         0
         0
```

下列範例會從常值值擷取毫秒。

```
select extract(ms from time '18:25:33.123456');

 date_part
-----------
     123
```

TIMETZ 範例

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ)，其中插入了三個值。

```
select timetz_val from timetz_test;

timetz_val
------------------
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會擷取每個 timetz_val 的小時。

```
select extract(hour from timetz_val) as hours from time_test;

hours
-----------
         4
         0
         5
```

下列範例會從常值值擷取毫秒。在處理擷取之前，常值不會轉換為 UTC。

```
select extract(ms from timetz '18:25:33.123456 EST');

 date_part
-----------
     123
```

下列範例會從常值 timetz 值傳回 UTC 的時區偏移小時。

```
select extract(timezone_hour from timetz '1.12.1997 07:37:16.00 PDT');

date_part
-----------
-7
```

具有間隔年到月和間隔日到第二的例子

下列範例會擷取定義 36 小時 (即 1 天 12 小時) 的1間隔天到秒的第二天部分。

```
select EXTRACT('days' from INTERVAL '36 hours' DAY TO SECOND)

 date_part
------------------
 1
```

下列範例會3從定義 15 個月 (即 1 年 3 個月) 的年份擷取月份部分。

```
select EXTRACT('month' from INTERVAL '15 months' YEAR TO MONTH)
```

```
 date_part
------------------
 3
```

下面的示例提取6從 30 個月，即 2 年 6 個月的月份部分。

```
select EXTRACT('month' from INTERVAL '30' MONTH)

 date_part
------------------
 6
```

下列範例會2從 50 小時 (即 2 天 2 小時) 擷取小時部分。

```
select EXTRACT('hours' from INTERVAL '50' HOUR)

 date_part
------------------
 2
```

下列範例會11從 1 小時 11 分鐘 11.123 秒擷取的分鐘部分。

```
select EXTRACT('minute' from INTERVAL '70 minutes 70.123 seconds' MINUTE TO SECOND)

 date_part
------------------
 11
```

下面的例子1.11從 1 天 1 小時 1 分鐘 1.11 秒提取秒部分。

```
select EXTRACT('seconds' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND)

 date_part
------------------
 1.11
```

下面的例子提取在一個 INTERVAL 的總小時數。每個零件都被萃取並加入到總計中。

```
select EXTRACT('days' from INTERVAL '50' HOUR) * 24 + EXTRACT('hours' from INTERVAL
 '50' HOUR)
```

```
 ?column?
------------------
 50
```

下列範例會擷取間隔中的總秒數。每個零件都被萃取並加入到總計中。

```
select EXTRACT('days' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 86400 +
       EXTRACT('hours' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 3600 +
       EXTRACT('minutes' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 60 +
       EXTRACT('seconds' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND)

 ?column?
------------------
 90061.11
```

GETDATE 函數

GETDATE 傳回目前工作階段時區中的目前日期和時間 (預設為 UTC)。它會傳回目前陳述式的開始日期或時間，即使是在交易區塊中也一樣。

語法

```
GETDATE()
```

括號是必要的。

傳回類型

TIMESTAMP

範例

下列範例中會使用 GETDATE 函數來傳回目前日期的完整時間戳記。

```
select getdate();

timestamp
---------------------
2008-12-04 16:10:43
```

下列範例中會使用 TRUNC 函數中的 GETDATE 函數來傳回不含時間的目前日期。

```
select trunc(getdate());

trunc
------------
2008-12-04
```

INTERVAL_CMP 函數

INTERVAL_CMP 會比較兩個間隔並傳回 1 (如果第一個間隔較大)、-1 (如果第二個間隔較大)，且 0 (如果間隔相等)。如需詳細資訊，請參閱 不含限定詞語法的間隔常值範例。

語法

```
INTERVAL_CMP(interval1, interval2)
```

引數

interval1

間隔常值。

interval2

間隔常值。

傳回類型

INTEGER

範例

下列範例會比較 3 days 和 1 year 的值。

```
select interval_cmp('3 days','1 year');

interval_cmp
--------------
-1
```

此範例會將值 7 days 與 1 week 進行比較。

```
select interval_cmp('7 days','1 week');

interval_cmp
--------------
0
```

下列範例會比較 1 year 和 3 days 的值。

```
select interval_cmp('1 year','3 days');

interval_cmp
--------------
1
```

LAST_DAY 函數

LAST_DAY 會傳回某月最後一天的日期，其中包含 date。不論 date 引數的資料類型為何，傳回類型一律是 DATE。

如需擷取特定日期部分的相關資訊，請參閱DATE_TRUNC 函數。

語法

```
LAST_DAY( { date | timestamp } )
```

引數

date | timestamp

資料類型 DATE 或 TIMESTAMP 的欄，或是隱含評估為 DATE 或 TIMESTAMP 類型的運算式。

傳回類型

DATE

範例

下列範例傳回目前月份之最後一天的日期。

```
select last_day(sysdate);
```

```
  last_day
------------
 2014-01-31
```

下列範例傳回該月份之最後 7 天每天所售的門票數。SALETIME 欄中的值是時間戳記。

```
select datediff(day, saletime, last_day(saletime)) as "Days Remaining", sum(qtysold)
from sales
where datediff(day, saletime, last_day(saletime)) < 7
group by 1
order by 1;

days remaining |  sum
---------------+-------
             0 | 10140
             1 | 11187
             2 | 11515
             3 | 11217
             4 | 11446
             5 | 11708
             6 | 10988
(7 rows)
```

MONTHS_BETWEEN 函數

MONTHS_BETWEEN 可判斷兩個日期之間有幾個月。

如果第一個日期晚於第二個日期，結果為正值，否則結果為負值。

如果引數為 null，則結果為 NULL。

語法

```
MONTHS_BETWEEN( date1, date2 )
```

引數

date1

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

date2

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

FLOAT8

結果的整個數字部分是基於日期的年份和月份值的差異。結果的小數部分是透過日期的天和時間戳記值計算而得，且一個月假設為 31 天。

如果 date1 和 date2 都含有某個月份中的相同日期 (例如，1/15/14 和 2/15/14) 或某月的最後一天 (例如，8/31/14 和 9/30/14)，則結果是根據日期之年份和月份值的整個數值，而不論時間戳記部分符合的情況 (若有)。

範例

下列範例傳回 1/18/1969 和 3/18/1969 的月份。

```
select months_between('1969-01-18', '1969-03-18')
as months;

months
----------
-2
```

下列範例傳回 1/18/1969 和 1/18/1969 的月份。

```
select months_between('1969-01-18', '1969-01-18')
as months;

months
----------
0
```

下列範例傳回事件之第一個和最後一個顯示間的月份。

```
select eventname,
min(starttime) as first_show,
max(starttime) as last_show,
months_between(max(starttime),min(starttime)) as month_diff
```

```
from event
group by eventname
order by eventname
limit 5;

eventname         first_show              last_show               month_diff
-------------------------------------------------------------------------------
.38 Special       2008-01-21 19:30:00.0  2008-12-25 15:00:00.0  11.12
3 Doors Down      2008-01-03 15:00:00.0  2008-12-01 19:30:00.0  10.94
70s Soul Jam      2008-01-16 19:30:00.0  2008-12-07 14:00:00.0  10.7
A Bronx Tale      2008-01-21 19:00:00.0  2008-12-15 15:00:00.0  10.8
A Catered Affair  2008-01-08 19:30:00.0  2008-12-19 19:00:00.0  11.35
```

NEXT_DAY 函數

NEXT_DAY 會傳回在給定日期之後的指定那天之第一個執行個體的日期。

如果 day 值與給定日期是一週中的同一天，則會傳回該日的下一個出現時間。

語法

```
NEXT_DAY( { date | timestamp }, day )
```

引數

date | timestamp

資料類型 DATE 或 TIMESTAMP 的欄，或是隱含評估為 DATE 或 TIMESTAMP 類型的運算式。

天

包含任何天之名稱的字串。大寫並不重要。

有效值如下。

天	值
週日	日、週日、星期日
週一	一、週一、星期一
週二	二、週二、星期二

天	值
週三	三、週三、星期三
週四	四、週四、星期四
週五	五、週五、星期五
週六	六、週六、星期六

傳回類型

DATE

範例

下列範例請求會傳回在 8/20/2014 後第一個星期二的日期。

```
select next_day('2014-08-20','Tuesday');

next_day
-----------
2014-08-26
```

下列範例會傳回 2008 年 1 月 1 日之後第一個星期二的 5:54:44 的日期。

```
select listtime, next_day(listtime, 'Tue') from listing limit 1;

listtime            | next_day
--------------------+-----------
2008-01-01 05:54:44 | 2008-01-08
```

以下範例會取得第三季目標行銷日期。

```
select username, (firstname ||' '|| lastname) as name,
eventname, caldate, next_day (caldate, 'Monday') as marketing_target
from sales, date, users, event
where sales.buyerid = users.userid
and sales.eventid = event.eventid
and event.dateid = date.dateid
and date.qtr = 3
```

```
order by marketing_target, eventname, name;

username  |     name          |     eventname        |    caldate   |
 marketing_target
----------+-------------------+----------------------+--------------
+-------------------
MBO26QSG  |   Callum Atkinson | .38 Special          |  2008-07-06  | 2008-07-07
WCR50YIU  |   Erasmus Alvarez | A Doll's House       |  2008-07-03  | 2008-07-07
CKT70OIE  |   Hadassah Adkins | Ana Gabriel          |  2008-07-06  | 2008-07-07
VVG07OUO  |   Nathan Abbott   | Armando Manzanero    |  2008-07-04  | 2008-07-07
GEW77SII  |   Scarlet Avila   | August: Osage County |  2008-07-06  | 2008-07-07
ECR71CVS  |   Caryn Adkins    | Ben Folds            |  2008-07-03  | 2008-07-07
KUW82CYU  |   Kaden Aguilar   | Bette Midler         |  2008-07-01  | 2008-07-07
WZE78DJZ  |   Kay Avila       | Bette Midler         |  2008-07-01  | 2008-07-07
HXY04NVE  |   Dante Austin    | Britney Spears       |  2008-07-02  | 2008-07-07
URY81YWF  |   Wilma Anthony   | Britney Spears       |  2008-07-02  | 2008-07-07
```

SYSDATE 函數

SYSDATE 傳回目前工作階段時區中的目前日期和時間 (預設為 UTC)。

> **Note**
>
> SYSDATE 傳回目前交易的日期和時間 (而不是目前陳述式的開始)。

語法

```
SYSDATE
```

此函數不需引數。

傳回類型

TIMESTAMP

範例

下列範例中會使用 SYSDATE 函數來傳回目前日期的完整時間戳記。

```
select sysdate;
```

```
timestamp
----------------------------
2008-12-04 16:10:43.976353
```

下列範例中會使用 TRUNC 函數中的 SYSDATE 函數來傳回不含時間的目前日期。

```
select trunc(sysdate);

trunc
------------
2008-12-04
```

在發出查詢且當日期早於 120 天時，下列查詢會傳回落於該日期間之日期的特價資訊。

```
select salesid, pricepaid, trunc(saletime) as saletime, trunc(sysdate) as now
from sales
where saletime between trunc(sysdate)-120 and trunc(sysdate)
order by saletime asc;

 salesid | pricepaid |  saletime  |    now
---------+-----------+------------+------------
91535    |    670.00 | 2008-08-07 | 2008-12-05
91635    |    365.00 | 2008-08-07 | 2008-12-05
91901    |   1002.00 | 2008-08-07 | 2008-12-05
...
```

TIMEOFDAY 函數

TIMEOFDAY 是特別的別名，會用來傳回週間日、日期和時間做為字串值。它會傳回目前陳述式的當日時間字串，即使是在交易區塊中也一樣。

語法

```
TIMEOFDAY()
```

傳回類型

VARCHAR

範例

下列範例會透過使用 TIMEOFDAY 函數來傳回目前日期和時間。

```
select timeofday();

timeofday
------------
Thu Sep 19 22:53:50.333525 2013 UTC
```

TIMESTAMP_CMP 函數

比較兩個時間戳記的值並傳回整數。如果時間戳記相同，則函數會傳回 0。如果第一個時間戳記較大，則函數會傳回 1。如果第二個時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMP_CMP(timestamp1, timestamp2)
```

引數

timestamp1

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

timestamp2

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較時間戳記，並顯示比較結果。

```
SELECT TIMESTAMP_CMP('2008-01-24 06:43:29', '2008-01-24 06:43:29'),
 TIMESTAMP_CMP('2008-01-24 06:43:29', '2008-02-18 02:36:48'), TIMESTAMP_CMP('2008-02-18
 02:36:48', '2008-01-24 06:43:29');

timestamp_cmp  | timestamp_cmp | timestamp_cmp
---------------+---------------+---------------
             0 |            -1 |             1
```

例如，下列範例會比較清單的 LISTTIME 與 SALETIME。所有清單的 TIMESTAMP_CMP 的值是 -1，因為特價的時間戳記是在清單的時間戳記後。

```
select listing.listid, listing.listtime,
sales.saletime, timestamp_cmp(listing.listtime, sales.saletime)
from listing, sales
where listing.listid=sales.listid
order by 1, 2, 3, 4
limit 10;

 listid |      listtime       |      saletime       | timestamp_cmp
--------+---------------------+---------------------+---------------
      1 | 2008-01-24 06:43:29 | 2008-02-18 02:36:48 |            -1
      4 | 2008-05-24 01:18:37 | 2008-06-06 05:00:16 |            -1
      5 | 2008-05-17 02:29:11 | 2008-06-06 08:26:17 |            -1
      5 | 2008-05-17 02:29:11 | 2008-06-09 08:38:52 |            -1
      6 | 2008-08-15 02:08:13 | 2008-08-31 09:17:02 |            -1
     10 | 2008-06-17 09:44:54 | 2008-06-26 12:56:06 |            -1
     10 | 2008-06-17 09:44:54 | 2008-07-10 02:12:36 |            -1
     10 | 2008-06-17 09:44:54 | 2008-07-16 11:59:24 |            -1
     10 | 2008-06-17 09:44:54 | 2008-07-22 02:23:17 |            -1
     12 | 2008-07-25 01:45:49 | 2008-08-04 03:06:36 |            -1
(10 rows)
```

此範例顯示若是在相同的時間戳記時，TIMESTAMP_CMP 會傳回 0：

```
select listid, timestamp_cmp(listtime, listtime)
from listing
order by 1 , 2
limit 10;

 listid | timestamp_cmp
--------+---------------
      1 |             0
      2 |             0
      3 |             0
      4 |             0
      5 |             0
      6 |             0
      7 |             0
      8 |             0
      9 |             0
     10 |             0
```

```
(10 rows)
```

TIMESTAMP_CMP_DATE 函數

TIMESTAMP_CMP_DATE 會比較時間戳記值和日期。如果時間戳記和日期值相同，則函數會傳回 0。如果時間戳記依時間順序較大，則函數會傳回 1。如果日期較大，則函數會傳回 -1。

語法

```
TIMESTAMP_CMP_DATE(timestamp, date)
```

引數

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

例如，下列範例會比較日期 2008-06-18 與 LISTTIME。在此日期後所做的清單會傳回 1，在此日期前所做的清單會傳回 -1。LISTTIME 值是時間戳記。

```
select listid, listtime,
timestamp_cmp_date(listtime, '2008-06-18')
from listing
order by 1, 2, 3
limit 10;


 listid |      listtime       | timestamp_cmp_date
--------+---------------------+--------------------
      1 | 2008-01-24 06:43:29 |                 -1
      2 | 2008-03-05 12:25:29 |                 -1
      3 | 2008-11-01 07:35:33 |                  1
```

```
      4 | 2008-05-24 01:18:37 |              -1
      5 | 2008-05-17 02:29:11 |              -1
      6 | 2008-08-15 02:08:13 |               1
      7 | 2008-11-15 09:38:15 |               1
      8 | 2008-11-09 05:07:30 |               1
      9 | 2008-09-09 08:03:36 |               1
     10 | 2008-06-17 09:44:54 |              -1
(10 rows)
```

TIMESTAMP_CMP_TIMESTAMPTZ 函數

TIMESTAMP_CMP_TIMESTAMPTZ 將時間戳記運算式值與含時區運算式的時間戳記比較。如果時間戳記和含時區值的時間戳記相同，則函數會傳回 0。如果時間戳記依時間順序較大，則函數會傳回 1。如果含時區的時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMP_CMP_TIMESTAMPTZ(timestamp, timestamptz)
```

引數

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較時間戳記與含時區的時間戳記，並顯示比較結果。

```
SELECT TIMESTAMP_CMP_TIMESTAMPTZ('2008-01-24 06:43:29', '2008-01-24 06:43:29+00'),
 TIMESTAMP_CMP_TIMESTAMPTZ('2008-01-24 06:43:29', '2008-02-18 02:36:48+00'),
 TIMESTAMP_CMP_TIMESTAMPTZ('2008-02-18 02:36:48', '2008-01-24 06:43:29+00');

timestamp_cmp_timestamptz  | timestamp_cmp_timestamptz | timestamp_cmp_timestamptz
---------------------------+---------------------------+--------------------------
```

```
        0           |          -1           |           1
```

TIMESTAMPTZ_CMP 函數

TIMESTAMPTZ_CMP 會比較含時區值的兩個時間戳記值並傳回整數。如果時間戳記相同，則函數會傳回 0。如果第一個時間戳記依時間順序較大，則函數會傳回 1。如果第二個時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMPTZ_CMP(timestamptz1, timestamptz2)
```

引數

timestamptz1

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

timestamptz2

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較含時區的時間戳記，並顯示比較結果。

```
SELECT TIMESTAMPTZ_CMP('2008-01-24 06:43:29+00', '2008-01-24 06:43:29+00'),
 TIMESTAMPTZ_CMP('2008-01-24 06:43:29+00', '2008-02-18 02:36:48+00'),
 TIMESTAMPTZ_CMP('2008-02-18 02:36:48+00', '2008-01-24 06:43:29+00');

timestamptz_cmp  | timestamptz_cmp | timestamptz_cmp
-----------------+-----------------+----------------
        0        |       -1        |       1
```

TIMESTAMPTZ_CMP_DATE 函數

TIMESTAMPTZ_CMP_DATE 會比較時間戳記值和日期的值。如果時間戳記和日期值相同，則函數會傳回 0。如果時間戳記依時間順序較大，則函數會傳回 1。如果日期較大，則函數會傳回 -1。

語法

```
TIMESTAMPTZ_CMP_DATE(timestamptz, date)
```

引數

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較做為含時區之時間戳記的 LISTTIME 與日期 2008-06-18。在此日期後所做的清單會傳回 1，在此日期前所做的清單會傳回 -1。

```
select listid, CAST(listtime as timestamptz) as tstz,
timestamp_cmp_date(tstz, '2008-06-18')
from listing
order by 1, 2, 3
limit 10;


 listid |          tstz          | timestamptz_cmp_date
--------+------------------------+----------------------
      1 | 2008-01-24 06:43:29+00 |                   -1
      2 | 2008-03-05 12:25:29+00 |                   -1
      3 | 2008-11-01 07:35:33+00 |                    1
      4 | 2008-05-24 01:18:37+00 |                   -1
      5 | 2008-05-17 02:29:11+00 |                   -1
      6 | 2008-08-15 02:08:13+00 |                    1
      7 | 2008-11-15 09:38:15+00 |                    1
      8 | 2008-11-09 05:07:30+00 |                    1
      9 | 2008-09-09 08:03:36+00 |                    1
     10 | 2008-06-17 09:44:54+00 |                   -1
(10 rows)
```

TIMESTAMPTZ_CMP_TIMESTAMP 函數

TIMESTAMPTZ_CMP_TIMESTAMP 將含時區運算式的時間戳記值與時間戳記運算式比較。如果含時區的時間戳記和時間戳記值相同，則函數會傳回 0。如果含時區的時間戳記依時間順序較大，則函數會傳回 1。如果時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMPTZ_CMP_TIMESTAMP(timestamptz, timestamp)
```

引數

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

傳回類型

INTEGER

範例

下列範例會將含時區的時間戳記與時間戳進行比較並顯示比較結果。

```
SELECT TIMESTAMPTZ_CMP_TIMESTAMP('2008-01-24 06:43:29+00', '2008-01-24 06:43:29'),
 TIMESTAMPTZ_CMP_TIMESTAMP('2008-01-24 06:43:29+00', '2008-02-18 02:36:48'),
 TIMESTAMPTZ_CMP_TIMESTAMP('2008-02-18 02:36:48+00', '2008-01-24 06:43:29');

timestamptz_cmp_timestamp  | timestamptz_cmp_timestamp | timestamptz_cmp_timestamp
---------------------------+---------------------------+--------------------------
             0             |            -1             |             1
```

TIMEZONE 函數

TIMEZONE 傳回時間戳記，做為指定時區和時間戳記值。

如需有關如何設定時區的資訊和範例，請參閱 timezone。

如需有關如何轉換時區的資訊和範例，請參閱 CONVERT_TIMEZONE。

語法

```
TIMEZONE('timezone', { timestamp | timestamptz })
```

引數

timezone

傳回值的時區。您可以將時區指定為時區名稱 (例如 **'Africa/Kampala'** 或 **'Singapore'**) 或做為時區縮寫 (例如 **'UTC'** 或 **'PDT'**)。若要查看受支援時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```

若要查看受支援時區縮寫的清單，請執行下列命令。

```
select pg_timezone_abbrevs();
```

如需詳細資訊和範例，請參閱 時區使用須知。

timestamp | timestamptz

造成 TIMESTAMP 或 TIMESTAMPTZ 類型的運算式會隱含地強制轉換為時間戳記或含時區的時間戳記。

傳回類型

與 TIMESTAMP 表達式搭配使用時的 TIMESTAMPTZ。

與 TIMESTAMPTZ 表達式搭配使用時的 TIMESTAMP。

範例

以下使用 PST 時區的時間戳記 2008-06-17 09:44:54 傳回 UTC 時區的時間戳記。

```
SELECT TIMEZONE('PST', '2008-06-17 09:44:54');

timezone
-----------------------
2008-06-17 17:44:54+00
```

以下使用 UTC 時區的時間戳記 2008-06-17 09:44:54+00 傳回 PST 時區的時間戳記。

```
SELECT TIMEZONE('PST', timestamptz('2008-06-17 09:44:54+00'));

timezone
-----------------------
2008-06-17 01:44:54
```

TO_TIMESTAMP 函數

TO_TIMESTAMP 會將 TIMESTAMP 字串轉換為 TIMESTAMPTZ。如需 Amazon Redshift 的其他日期和時間函數清單，請參閱日期和時間函數。

語法

```
to_timestamp(timestamp, format)
```

```
to_timestamp (timestamp, format, is_strict)
```

引數

timestamp

字串，代表 format 指定的格式中的時間戳記值。如果此引數保留為空白，則時間戳記值預設為 `0001-01-01 00:00:00`。

format

字串常值，定義 timestamp 值的格式。不支援包含時區 (**TZ**、**tz** 或 **OF**) 的格式做為輸出。請參閱日期時間格式字串 以取得有效的時間戳記格式。

is_strict

選用的 Boolean 值，指定如果輸入時間戳值超出範圍是否回傳錯誤。當 is_strict 設定為 TRUE 時，如果有超出範圍的值，就會傳回錯誤。當 is_strict 設定為 FALSE (預設值) 時，會接受溢位值。

傳回類型

TIMESTAMPTZ

範例

以下範例示範如何使用 TO_TIMESTAMP 函數，將 TIMESTAMP 字串轉換成 TIMESTAMPTZ。

```
select sysdate, to_timestamp(sysdate, 'YYYY-MM-DD HH24:MI:SS') as second;

timestamp                  | second
---------------------------   ---------------------
2021-04-05 19:27:53.281812 | 2021-04-05 19:27:53+00
```

可以傳遞日期的 TO_TIMESTAMP 部分。其餘日期部分設定為預設值。時間包含在輸出中：

```
SELECT TO_TIMESTAMP('2017','YYYY');

to_timestamp
--------------------------
2017-01-01 00:00:00+00
```

以下 SQL 陳述式將字串 '2011-12-18 24:38:15' 轉換為 TIMESTAMPTZ。結果是 TIMESTAMPTZ 落在第二天，因為小時數超過 24 小時：

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS');

to_timestamp
---------------------
2011-12-19 00:38:15+00
```

下列 SQL 陳述式會將字串「2011-12-18 24:38:15」轉換為 TIMESTAMPTZ。結果會產生錯誤，因為時間戳記中的時間值超過 24 小時：

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS', TRUE);

ERROR:  date/time field time value out of range: 24:38:15.0
```

TRUNC 函數

截斷 TIMESTAMP 並傳回 DATE。

這個函數也可以截斷數字。如需詳細資訊，請參閱 TRUNC 函數。

語法

```
TRUNC(timestamp)
```

引數

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

若要傳回以 00:00:00 為時間的時間戳記值，請將函數結果轉換為 TIMESTAMP。

傳回類型

DATE

範例

以下範例從 SYSDATE 函數的結果傳回日期部分 (其會傳回時間戳記)。

```
SELECT SYSDATE;

+----------------------------+
|         timestamp          |
+----------------------------+
| 2011-07-21 10:32:38.248109 |
+----------------------------+

SELECT TRUNC(SYSDATE);

+------------+
|   trunc    |
+------------+
| 2011-07-21 |
+------------+
```

下列範例會將 TRUNC 函數套用至 TIMESTAMP 欄。傳回類型為日期。

```
SELECT TRUNC(starttime) FROM event
ORDER BY eventid LIMIT 1;

+------------+
|   trunc    |
+------------+
| 2008-01-25 |
+------------+
```

下列範例會傳回時間為 00:00:00 的時間戳記值，方法是將 TRUNC 函數結果轉換 TIMESTAMP。

```
SELECT CAST((TRUNC(SYSDATE)) AS TIMESTAMP);

+---------------------+
|        trunc        |
+---------------------+
| 2011-07-21 00:00:00 |
+---------------------+
```

日期或時間戳記函數的日期部分

下表識別日期部分和時間部分名稱和縮寫，系統接受它們做為以下函數的引數：

- DATEADD
- DATEDIFF
- DATE_PART
- EXTRACT

日期部分或時間部分	縮寫
millennium, millennia	mil, mils
century, centuries	c, cent, cents
decade, decades	dec, decs
epoch	epoch (由 EXTRACT 支援)
year, years	y, yr, yrs
quarter, quarters	qtr, qtrs
month, months	mon, mons
week, weeks	w
週中的日	dayofweek, dow, dw, weekday (由 DATE_PART 和 EXTRACT 函數 所支援)

日期部分或時間部分	縮寫
	傳回從 0–6 的整數，從星期日開始。 **Note** DOW 日期部分與週中的日 (日期部分用來表示日期時間格是字串) 表現不同 (D)。D 是基於整數 1–7，其中星期日為 1。如需詳細資訊，請參閱 日期時間格式字串。
年中的日	dayofyear, doy, dy, yearday (由 EXTRACT 支援)
day, days	d
hour, hours	h, hr, hrs
minute, minutes	m, min, mins
second, seconds	s, sec, secs
millisecond, milliseconds	ms, msec, msecs, msecond, mseconds, millisec, millisecs, millisecon
microsecond, microseconds	microsec, microsecs, microsecond, usecond, useconds, us, usec, usecs
timezone, timezone_hour, timezone_minute	由 EXTRACT 函數僅針對含時區 (TIMESTAMPTZ) 的時間戳記所支援。

含秒、毫秒和微秒結果的差異

當不同日期函數指定秒、毫秒或微秒做為日期部分時，減去查詢結果的差異：

- EXTRACT 函數會傳回整數僅做為指定日期部分，而忽略更高或更低層級的日期部分。若指定的日期部分為秒，則毫秒或微秒不會包含在結果中。若指定的日期部分為豪秒，則秒或微秒不會包含在內。若指定的日期部分為微秒，則秒或豪秒不會包含在內。
- DATE_PART 函數會傳回時間戳記的完整秒部分，不論指定的日期部分為何，視需要傳回十進位值或整數。

例如，比較以下查詢的結果：

```
create table seconds(micro timestamp);

insert into seconds values('2009-09-21 11:10:03.189717');

select extract(sec from micro) from seconds;

date_part
-----------
3

select date_part(sec, micro) from seconds;

pgdate_part
-------------
3.189717
```

CENTURY、EPOCH、DECADE 和 MIL 備註

CENTURY 或 CENTURIES

Amazon Redshift 會解釋 CENTURY 來以年 ###1 開始並以年 ###0 結束：

```
select extract (century from timestamp '2000-12-16 12:21:13');
date_part
-----------
20

select extract (century from timestamp '2001-12-16 12:21:13');
date_part
-----------
21
```

EPOCH

EPOCH 的 Amazon Redshift 實作相對於 1970-01-01 00:00:00.000000，與叢集所在的時區無關。您可能想要根據叢集所在的時區，彌補結果的時數差異。

下列範例示範以下方法：

1. 建立名為 EVENT_EXAMPLE 且根據 EVENT 資料欄的資料表。此 CREATE AS 命令會使用 DATE_PART 函數來建立日期欄 (依預設，名為 PGDATE_PART)，以儲存每個事件的 epoch 值。
2. 從 PG_TABLE_DEF 選取 EVENT_EXAMPLE 的欄和資料類型。
3. 從 EVENT_EXAMPLE 資料表選取 EVENTNAME、STARTTIME 和 PGDATE_PART，來檢視不同的日期和時間格式。
4. 從 EVENT EXAMPLE 中按現狀選取 EVENTNAME 和 STARTTIME。使用 1 秒間隔將 PGDATE_PART 中的 epoch 值轉換為不含時區的時間戳記，並在名為 CONVERTED_TIMESTAMP 的欄位中傳回結果。

```
create table event_example
as select eventname, starttime, date_part(epoch, starttime) from event;

select "column", type from pg_table_def where tablename='event_example';

     column    |            type
---------------+-----------------------------
 eventname     | character varying(200)
 starttime     | timestamp without time zone
 pgdate_part   | double precision
(3 rows)
```

```
select eventname, starttime, pgdate_part from event_example;

   eventname          |      starttime      | pgdate_part
----------------------+---------------------+-------------
 Mamma Mia!           | 2008-01-01 20:00:00 |  1199217600
 Spring Awakening     | 2008-01-01 15:00:00 |  1199199600
 Nas                  | 2008-01-01 14:30:00 |  1199197800
 Hannah Montana       | 2008-01-01 19:30:00 |  1199215800
 K.D. Lang            | 2008-01-01 15:00:00 |  1199199600
 Spamalot             | 2008-01-02 20:00:00 |  1199304000
 Macbeth              | 2008-01-02 15:00:00 |  1199286000
 The Cherry Orchard   | 2008-01-02 14:30:00 |  1199284200
 Macbeth              | 2008-01-02 19:30:00 |  1199302200
 Demi Lovato          | 2008-01-02 19:30:00 |  1199302200


select eventname,
starttime,
```

```
timestamp with time zone 'epoch' + pgdate_part * interval '1 second' AS
 converted_timestamp
from event_example;

       eventname      |      starttime      | converted_timestamp
----------------------+---------------------+---------------------
 Mamma Mia!           | 2008-01-01 20:00:00 | 2008-01-01 20:00:00
 Spring Awakening     | 2008-01-01 15:00:00 | 2008-01-01 15:00:00
 Nas                  | 2008-01-01 14:30:00 | 2008-01-01 14:30:00
 Hannah Montana       | 2008-01-01 19:30:00 | 2008-01-01 19:30:00
 K.D. Lang            | 2008-01-01 15:00:00 | 2008-01-01 15:00:00
 Spamalot             | 2008-01-02 20:00:00 | 2008-01-02 20:00:00
 Macbeth              | 2008-01-02 15:00:00 | 2008-01-02 15:00:00
 The Cherry Orchard   | 2008-01-02 14:30:00 | 2008-01-02 14:30:00
 Macbeth              | 2008-01-02 19:30:00 | 2008-01-02 19:30:00
 Demi Lovato          | 2008-01-02 19:30:00 | 2008-01-02 19:30:00
 ...
```

DECADE 或 DECADES

Amazon Redshift 會根據一般日曆解釋 DECADE 或 DECADES DATEPART。例如，因為一般日曆是從年份 1 開始，第一個十年 (十年 1) 為 0001-01-01 至 0009-12-31，而第二個十年 (十年 2) 為 0010-01-01 到 0019-12-31。例如，十年 201 橫跨 2000-01-01 到 2009-12-31：

```
select extract(decade from timestamp '1999-02-16 20:38:40');
date_part
-----------
200

select extract(decade from timestamp '2000-02-16 20:38:40');
date_part
-----------
201

select extract(decade from timestamp '2010-02-16 20:38:40');
date_part
-----------
202
```

MIL 或 MILS

Amazon Redshift 會解釋 MIL 來以年 #001 的第一天開始並以年 #000 的最後一天結束：

```
select extract (mil from timestamp '2000-12-16 12:21:13');
date_part
-----------
2

select extract (mil from timestamp '2001-12-16 12:21:13');
date_part
-----------
3
```

雜湊函數

主題

- CHECKSUM 函數
- farmFingerprint64 函數
- FUNC_SHA1 函數
- FNV_HASH 函數
- MD5 函數
- SHA 函數
- SHA1 函數
- SHA2 函數
- MURMUR3_32_HASH

雜湊函數是一種數學函數，可將數字輸入值轉換成另一個值。

CHECKSUM 函數

計算用來建立雜湊索引的檢查總和值。

語法

```
CHECKSUM(expression)
```

引數

運算式

輸入表達式必須為 VARCHAR、INTEGER 或 DECIMAL 資料類型。

傳回類型

CHECKSUM 函數傳回整數。

範例

下列範例計算 COMMISSION 欄的檢查總和值：

```
select checksum(commission)
from sales
order by salesid
limit 10;

checksum
----------
10920
1140
5250
2625
2310
5910
11820
2955
8865
975
(10 rows)
```

farmFingerprint64 函數

使用 `Fingerprint64` 函數計算輸入引數的 farmhash 值。

語法

```
farmFingerprint64(expression)
```

引數

運算式

輸入運算式必須是 VARCHAR 或 VARBYTE 資料類型。

傳回類型

farmFingerprint64 函數會傳回 BIGINT。

範例

下列範例會傳回輸入 Amazon Redshift 為 VARCHAR 資料類型之 的 farmFingerprint64 值。

```
SELECT farmFingerprint64('Amazon Redshift');
```

```
  farmfingerprint64
---------------------
 8085098817162212970
```

下列範例會傳回輸入 Amazon Redshift 為 VARBYTE 資料類型之 的 farmFingerprint64 值。

```
SELECT farmFingerprint64('Amazon Redshift'::varbyte);
```

```
  farmfingerprint64
---------------------
 8085098817162212970
```

FUNC_SHA1 函數

SHA1 函數的同義詞。

請參閱SHA1 函數。

FNV_HASH 函數

針對所有基本資料類型運算 64 位元 FNV-1a 非密碼編譯雜湊函數。

語法

```
FNV_HASH(value [, seed])
```

引數

值

要進行雜湊的輸入值。Amazon Redshift 使用值的二進位表示法來雜湊輸入值；例如：INTEGER 值會使用 4 位元組雜湊，BIGINT 值則會使用 8 位元組雜湊。此外，CHAR 和 VARCHAR 輸入不會忽略結尾空格。

seed

雜湊函數的 BIGINT 種子是選用的。如果沒有指定，Amazon Redshift 會使用預設 FNV 種子。這會允許組合多個欄位的雜湊，而無須進行轉換或串連。

傳回類型

BIGINT

範例

下列範例會分別傳回數字、字串 'Amazon Redshift' 及兩個值串連之後的 FNV 雜湊。

```
select fnv_hash(1);
        fnv_hash
----------------------
 -5968735742475085980
(1 row)
```

```
select fnv_hash('Amazon Redshift');
      fnv_hash
---------------------
 7783490368944507294
(1 row)
```

```
select fnv_hash('Amazon Redshift', fnv_hash(1));
       fnv_hash
----------------------
 -2202602717770968555
```

```
(1 row)
```

使用須知

- 如要運算包含多個欄位資料表的雜湊，您可以運算第一個欄位的 FNV 雜湊，然後將其做為種子傳遞給第二個欄位的雜湊。然後，其會將第二個欄位的雜湊做為種子傳遞至第三個欄位的雜湊。

  以下範例會建立種子來雜湊包含多個欄位的資料表。

  ```
  select fnv_hash(column_3, fnv_hash(column_2, fnv_hash(column_1))) from sample_table;
  ```

- 相同屬性可以用來運算字串串連的雜湊。

  ```
  select fnv_hash('abcd');
         fnv_hash
  ---------------------
   -281581062704388899
  (1 row)
  ```

  ```
  select fnv_hash('cd', fnv_hash('ab'));
        fnv_hash
  ---------------------
   -281581062704388899
  (1 row)
  ```

- 雜湊函數會使用輸入的類型來判斷要雜湊的位元組數。如有必要，其會使用轉換來強制使用特定類型。

  以下範例使用不同的輸入類型來產生不同結果。

  ```
  select fnv_hash(1::smallint);
        fnv_hash
  --------------------
   589727492704079044
  (1 row)
  ```

  ```
  select fnv_hash(1);
         fnv_hash
  ----------------------
   -5968735742475085980
  ```

```
(1 row)
```

```
select fnv_hash(1::bigint);
       fnv_hash
----------------------
 -8517097267634966620
(1 row)
```

MD5 函數

使用 MD5 加密雜湊函數，將可變長度字串轉換為 32 的字元的字串，此為 128 位元檢查總和之十六進位值的文字表示法。

語法

```
MD5(string)
```

引數

string

可變長度字串。

傳回類型

MD5 函數傳回 32 個字元的字串，此為 128 位元檢查總和之十六進位值的文字表示法。

範例

下列範例顯示字串 'Amazon Redshift' 的 128 位元值：

```
select md5('Amazon Redshift');
md5
----------------------------------
f7415e33f972c03abd4f3fed36748f7a
(1 row)
```

SHA 函數

SHA1 函數的同義詞。

請參閱SHA1 函數。

SHA1 函數

SHA1 函數使用 SHA1 加密雜湊函數，將可變長度字串轉換為 40 之字元的字串，即 160 位元檢查總和之十六進位值的文字表示法。

語法

SHA1 是 SHA 函數 及 FUNC_SHA1 函數 的同義詞。

```
SHA1(string)
```

引數

string

可變長度字串。

傳回類型

SHA1 函數傳回 40 個字元的字串，此為 160 位元檢查總和之十六進位值的文字表示法。

範例

下列範例傳回字詞 'Amazon Redshift' 的 160 位元值：

```
select sha1('Amazon Redshift');
```

SHA2 函數

SHA2 函數使用 SHA2 加密雜湊函數，將可變長度字串轉換為一個字元的字串。字串是檢查總和之十六進位值的文字表示法，具有指定的位元數。

語法

```
SHA2(string, bits)
```

引數

string

可變長度字串。

integer

雜湊函數中的位元數。有效值為 0 (與 256 相同)、224、256、384 和 512。

傳回類型

SHA2 函數傳回一個字元的字串，它是檢查總和之十六進位值的文字表示法，或是空字串 (如果位元數無效)。

範例

下列範例傳回字詞 'Amazon Redshift' 的 256 位元值：

```
select sha2('Amazon Redshift', 256);
```

MURMUR3_32_HASH

MURMUR3_32_HASH 函數會計算所有常見資料類型 (包括數字和字串類型) 的 32 位元 Murmur3A 非加密雜湊。

語法

```
MURMUR3_32_HASH(value [, seed])
```

引數

值

要進行雜湊的輸入值。Amazon Redshift 雜湊輸入值的二進位表示。這種行為類似於 FNV_HASH 函數，但該值被轉換為 Apache Iceberg 32 位元 Murmur3 雜湊規格指定的二進位表示。

seed

雜湊函數的 INT 種子。此為選用引數。如果未指定，Amazon Redshift 將使用預設種子 0。這會允許組合多個欄位的雜湊，而無須進行轉換或串連。

傳回類型

函數會傳回 INT。

範例

下列範例會分別傳回數字、字串 'Amazon Redshift' 及兩個值串連之後的 Murmur3 雜湊。

```
select MURMUR3_32_HASH(1);

    MURMUR3_32_HASH
----------------------
 -5968735742475085980
(1 row)
```

```
select MURMUR3_32_HASH('Amazon Redshift');

    MURMUR3_32_HASH
----------------------
 7783490368944507294
(1 row)
```

```
select MURMUR3_32_HASH('Amazon Redshift', MURMUR3_32_HASH(1));

    MURMUR3_32_HASH
----------------------
 -2202602717770968555
(1 row)
```

使用須知

如要運算包含多個欄位資料表的雜湊，您可以運算第一個欄位的 Murmur3 雜湊，然後將其做為種子傳遞給第二個欄位的雜湊。然後，其會將第二個欄位的 Murmur3 雜湊做為種子傳遞至第三個欄位的雜湊。

以下範例會建立種子來雜湊包含多個欄位的資料表。

```
select MURMUR3_32_HASH(column_3, MURMUR3_32_HASH(column_2, MURMUR3_32_HASH(column_1)))
 from sample_table;
```

相同屬性可以用來運算字串串連的雜湊。

```
select MURMUR3_32_HASH('abcd');

   MURMUR3_32_HASH
---------------------
 -281581062704388899
(1 row)
```

```
select MURMUR3_32_HASH('cd', MURMUR3_32_HASH('ab'));

   MURMUR3_32_HASH
---------------------
 -281581062704388899
(1 row)
```

雜湊函數會使用輸入的類型來判斷要雜湊的位元組數。如有必要，其會使用轉換來強制使用特定類型。

以下範例使用不同的輸入類型來產生不同結果。

```
select MURMUR3_32_HASH(1::smallint);

   MURMUR3_32_HASH
--------------------
 589727492704079044
(1 row)
```

```
select MURMUR3_32_HASH(1);

   MURMUR3_32_HASH
----------------------
 -5968735742475085980
(1 row)
```

```
select MURMUR3_32_HASH(1::bigint);

   MURMUR3_32_HASH
----------------------
 -8517097267634966620
(1 row)
```

HyperLogLog 函數

接下來，您可以找到有關 Amazon Redshift 支持的 SQL HyperLogLog 函數的說明。

主題

HLL 函數

HLL 函數返回輸入表達 HyperLogLog 式值的基數。HLL 函數適用於除 HLLSKETCH 資料類型以外的任何資料類型。HLL 函數忽略 NULL 值。當資料表中沒有資料列或所有資料列都是 NULL 時，產生的基數為 0。

語法

```
HLL (aggregate_expression)
```

引數

aggregate_expression

提供要彙總之值的任何有效運算式，例如欄名。此函數支援除 HLLSKETCH、GEOMETRY、GEOGRAPHY 和 VARBYTE 以外的任何資料類型作為輸入。

傳回類型

HLL 函數會傳回 BIGINT 或 INT8 值。

範例

下列範例會傳回資料表 a_table 中資料欄 an_int 的基數。

```
CREATE TABLE a_table(an_int INT);
INSERT INTO a_table VALUES (1), (2), (3), (4);
```

```
SELECT hll(an_int) AS cardinality FROM a_table;
cardinality
-------------
4
```

HLL_CREATE_SKETCH 函數

HLL_CREATE_SKETCH 函數傳回封裝輸入運算式值的 HLLSKETCH 資料類型。HLL_CREATE_SKETCH 函數適用於任何資料類型，並忽略 NULL 值。如果表格中沒有列或所有列都為 NULL，則產生的草圖沒有索引-值對，例如 `{"version":1,"logm":15,"sparse":{"indices":[],"values":[]}}`。

語法

```
HLL_CREATE_SKETCH (aggregate_expression)
```

引數

aggregate_expression

提供要彙總之值的任何有效運算式，例如欄名。NULL 值將被忽略。此函數支援除 HLLSKETCH、GEOMETRY、GEOGRAPHY 和 VARBYTE 以外的任何資料類型作為輸入。

傳回類型

HLL_CREATE_SKETCH 函數會傳回 HLLSKETCH 值。

範例

下列範例會傳回資料表 `a_table` 中資料欄 `an_int` 的 HLLSKETCH 類型。匯入、匯出或列印草圖時，JSON 物件可用來表示稀疏 HyperLogLog草圖。字串表示 (採用 Base64 格式) 用於表示密集 HyperLogLog 草圖。

```
CREATE TABLE a_table(an_int INT);
INSERT INTO a_table VALUES (1), (2), (3), (4);

SELECT hll_create_sketch(an_int) AS sketch FROM a_table;
sketch
-----------------------------------------------------------------------------------------------------
{"version":1,"logm":15,"sparse":{"indices":
[20812342,20850007,22362299,47158030],"values":[1,2,1,1]}}
```

```
(1 row)
```

HLL_CARDINALITY 函數

HLL_CARDINALITY 函數會傳回輸入 HLLSKETCH 資料類型的基數。

語法

```
HLL_CARDINALITY (hllsketch_expression)
```

引數

hllsketch_expression

任何演算為 HLLSKETCH 類型的有效運算式，例如欄名稱。輸入值是 HLLSKETCH 資料類型。

傳回類型

HLL_CARDINALITY 函數會傳回 BIGINT 或 INT8 值。

範例

下列範例會傳回資料表 hll_table 中資料欄 sketch 的基數。

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
 (5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

SELECT hll_cardinality(sketch) AS cardinality FROM hll_table;
cardinality
-------------
6
4
(2 rows)
```

HLL_COMBINE 函數

HLL_COMBINE 彙總函數會傳回合併所有輸入 HLLSKETCH 值的 HLLSKETCH 資料類型。

兩個或多個 HyperLogLog 草圖的組合是一個新的 HLLSKETCH，它封裝了有關每個輸入草圖所代表的不同值聯集的信息。在合併草圖之後，Amazon Redshift 會擷取兩個或多個資料集的聯集基數。如需如何合併多個草圖的相關資訊，請參閱範例：從合併多個草圖返回草圖 HyperLogLog 。

語法

```
HLL_COMBINE (hllsketch_expression)
```

引數

hllsketch_expression

任何演算為 HLLSKETCH 類型的有效運算式，例如欄名稱。輸入值是 HLLSKETCH 資料類型。

傳回類型

HLL_COMBINE 函數會傳回 HLLSKETCH 類型。

範例

下列範例會傳回資料表 hll_table 中的合併 HLLSKETCH 值。

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
 (5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

SELECT hll_combine(sketch) AS sketches FROM hll_table;
sketches
--------------------------------------------------------------------------------------------
{"version":1,"logm":15,"sparse":{"indices":
[20812342,20850007,22362299,40314817,42650774,47158030],"values":[1,2,1,3,2,1]}}
(1 row)
```

HLL_COMBINE_SKETCHES 函數

HLL_COMBINE_SKETCHES 是純量函數，用作輸入兩個 HLLSKETCH 值，並將它們合併為一個 HLLSKETCH。

兩個或多個 HyperLogLog 草圖的組合是一個新的 HLLSKETCH，它封裝了有關每個輸入草圖所代表的不同值聯集的信息。

語法

```
HLL_COMBINE_SKETCHES (hllsketch_expression1, hllsketch_expression2)
```

引數

hllsketch_expression1 和 hllsketch_expression2

任何演算為 HLLSKETCH 類型的有效運算式，例如欄名稱。

傳回類型

HLL_COMBINE_SKETCHES 函數會傳回 HLLSKETCH 類型。

範例

下列範例會傳回資料表 hll_table 中的合併 HLLSKETCH 值。

```
WITH tbl1(x, y)
     AS (SELECT Hll_create_sketch(1),
                Hll_create_sketch(2)
         UNION ALL
         SELECT Hll_create_sketch(3),
                Hll_create_sketch(4)
         UNION ALL
         SELECT Hll_create_sketch(5),
                Hll_create_sketch(6)
         UNION ALL
         SELECT Hll_create_sketch(7),
                Hll_create_sketch(8)),
     tbl2(x, y)
     AS (SELECT Hll_create_sketch(9),
                Hll_create_sketch(10)
         UNION ALL
         SELECT Hll_create_sketch(11),
                Hll_create_sketch(12)
         UNION ALL
         SELECT Hll_create_sketch(13),
                Hll_create_sketch(14)
         UNION ALL
```

```
        SELECT Hll_create_sketch(15),
               Hll_create_sketch(16)
        UNION ALL
        SELECT Hll_create_sketch(NULL),
               Hll_create_sketch(NULL)),
     tbl3(x, y)
     AS (SELECT *
         FROM   tbl1
         UNION ALL
         SELECT *
         FROM   tbl2)
SELECT Hll_combine_sketches(x, y)
FROM   tbl3;
```

JSON 函數

主題

- IS_VALID_JSON 函數
- IS_VALID_JSON_ARRAY 函數
- JSON_ARRAY_LENGTH 函數
- JSON_EXTRACT_ARRAY_ELEMENT_TEXT 函數
- JSON_EXTRACT_PATH_TEXT 函數
- JSON_PARSE 函數
- CAN_JSON_PARSE 函數
- JSON_SERIALIZE 函數
- JSON_SERIALIZE_TO_VARBYTE 函數

當您需要儲存相當小的一組金鑰值對時，您可以用 JSON 格式儲存資料以節省空間。因為 JSON 字串可儲存於單一欄，採用 JSON 可能比以資料表格式儲存資料更有效率。例如，假設您有一個稀疏資料表，且您需要有許多欄才能完整代表所有可能的屬性，但任何給定列或任何給定欄的大部分欄值都是 NULL。如果採用 JSON 來儲存，您可以在單一 JSON 字串中以金鑰:值對來儲存列的資料，避免產生稀疏填入的資料表欄。

此外，您可以輕鬆修改 JSON 字串來儲存其他金鑰:值對，而不需要在資料表中新增欄。

建議少用 JSON。JSON 不適合儲存較大的資料集，因為 JSON 將相異的資料儲存在單一欄，未使用 Amazon Redshift 的欄儲存架構。雖然 Amazon Redshift 在 CHAR 和 VARCHAR 欄上支援 JSON 函

數，但我們建議您使用 SUPER 來處理 JSON 序列化格式的資料。SUPER 使用可以有效地查詢階層式資料的剖析後無結構描述表示。如需 SUPER 資料類型的相關資訊，請參閱 在 Amazon Redshift 中擷取和查詢半結構化資料。

JSON 使用 UTF-8 編碼的文字字串，所以 JSON 字串可儲存為 CHAR 或 VARCHAR 資料類型。如果字串包含多位元組字元，請使用 VARCHAR。

JSON 字串必須是符合下列規則的適當格式化 JSON：

- 根層級 JSON 可以是 JSON 物件或 JSON 陣列。JSON 物件是一組未排序的逗號分隔金鑰:值對 (以大括號括住)。

  例如：`{"one":1, "two":2}`
- JSON 陣列是一組已排序的逗號分隔值 (以方括號括住)。

  以下是範例：`["first", {"one":1}, "second", 3, null]`
- JSON 陣列採用以零開始的索引；陣列的第一個元素在位置 0。在 JSON 金鑰:值對中，金鑰是雙引號括住的字串。
- JSON 值可以是下列任何值：
  - JSON 物件
  - JSON 陣列
  - 以雙引號括住的字串
  - 數字 (整數和浮點數)
  - 布林值
  - null
- 空物件和空陣列是有效的 JSON 值。
- JSON 欄位區分大小寫。
- 忽略 JSON 結構元素之間的空格 (例如 `{ }, [ ]`)。

Amazon Redshift JSON 函數和 Amazon Redshift COPY 命令使用相同方法來處理 JSON 格式的資料。如需使用 JSON 的相關資訊，請參閱從 JSON 格式 COPY

IS_VALID_JSON 函數

IS_VALID_JSON 函數會驗證 JSON 字串。如果字串是格式正確的 JSON，此函數會傳回布林值 `true`，如果字串的格式不正確，則傳回 `false`。若要驗證 JSON 陣列，請使用 IS_VALID_JSON_ARRAY 函數

如需詳細資訊，請參閱 JSON 函數。

語法

```
IS_VALID_JSON('json_string')
```

引數

json_string

評估為 JSON 字串的字串或表達式。

傳回類型

BOOLEAN

範例

若要建立資料表並插入 JSON 字串來測試，請使用下列範例。

```
CREATE TABLE test_json(id int IDENTITY(0,1), json_strings VARCHAR);

-- Insert valid JSON strings --
INSERT INTO test_json(json_strings) VALUES
('{"a":2}'),
('{"a":{"b":{"c":1}}}'),
('{"a": [1,2,"b"]}');

-- Insert invalid JSON strings --
INSERT INTO test_json(json_strings) VALUES
('{{}}'),
('{1:"a"}'),
('[1,2,3]');
```

若要驗證上述範例中的字串，請使用下列範例。

```
SELECT id, json_strings, IS_VALID_JSON(json_strings)
FROM test_json
ORDER BY id;

+----+--------------------+--------------+
| id |    json_strings    | is_valid_json |
+----+--------------------+--------------+
|  0 | {"a":2}            | true         |
|  4 | {"a":{"b":{"c":1}}} | true         |
|  8 | {"a": [1,2,"b"]}   | true         |
| 12 | {{}}               | false        |
| 16 | {1:"a"}            | false        |
| 20 | [1,2,3]            | false        |
+----+--------------------+--------------+
```

IS_VALID_JSON_ARRAY 函數

IS_VALID_JSON_ARRAY 函數可驗證 JSON 陣列。如果陣列是格式正確的 JSON，此函數會傳回布林值 `true`，如果陣列的格式不正確，則傳回 `false`。若要驗證 JSON 字串，請使用 IS_VALID_JSON 函數

如需詳細資訊，請參閱 JSON 函數。

語法

```
IS_VALID_JSON_ARRAY('json_array')
```

引數

json_array

評估為 JSON 陣列的字串或表達式。

傳回類型

BOOLEAN

範例

若要建立資料表並插入 JSON 字串來測試，請使用下列範例。

```
CREATE TABLE test_json_arrays(id int IDENTITY(0,1), json_arrays VARCHAR);

-- Insert valid JSON array strings --
INSERT INTO test_json_arrays(json_arrays)
VALUES('[]'),
('["a","b"]'),
('["a",["b",1,["c",2,3,null]]]');

-- Insert invalid JSON array strings --
INSERT INTO test_json_arrays(json_arrays)
VALUES('{"a":1}'),
('a'),
('[1,2,]');
```

若要驗證上述範例中的字串，請使用下列範例。

```
SELECT json_arrays, IS_VALID_JSON_ARRAY(json_arrays)
FROM test_json_arrays ORDER BY id;

+------------------------------+---------------------+
|         json_arrays          | is_valid_json_array |
+------------------------------+---------------------+
| []                           | true                |
| ["a","b"]                    | true                |
| ["a",["b",1,["c",2,3,null]]] | true                |
| {"a":1}                      | false               |
| a                            | false               |
| [1,2,]                       | false               |
+------------------------------+---------------------+
```

JSON_ARRAY_LENGTH 函數

JSON_ARRAY_LENGTH 函數傳回 JSON 字串外圍陣列中的元素個數。如果 null_if_invalid 引數設為 `true`，且 JSON 字串無效，此函數會傳回 NULL，而非傳回錯誤。

如需詳細資訊，請參閱 JSON 函數。

語法

```
JSON_ARRAY_LENGTH('json_array' [, null_if_invalid ] )
```

引數

json_array

格式正確的 JSON 陣列。

null_if_invalid

(選用) BOOLEAN 值，指定輸入 JSON 字串無效時是否傳回 NULL，而非傳回錯誤。若要在 JSON 無效時傳回 NULL，請指定 `true` (t)。若要在 JSON 無效時傳回錯誤，請指定 `false` (f)。預設值為 `false`。

傳回類型

`INTEGER`

範例

若要傳回陣列中的元素個數，請使用下列範例。

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14]');

+-------------------+
| json_array_length |
+-------------------+
|                 5 |
+-------------------+
```

若要因 JSON 無效而傳回錯誤，請使用下列範例。

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14');

ERROR: invalid json array object [11,12,13,{"f1":21,"f2":[25,26]},14
```

若要將 null_if_invalid 設為 true，以便在 JSON 無效時，陳述式傳回 NULL 而非傳回錯誤，請使用下列範例。

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14',true);

+-------------------+
| json_array_length |
+-------------------+
```

```
| NULL               |
+--------------------+
```

JSON_EXTRACT_ARRAY_ELEMENT_TEXT 函數

JSON_EXTRACT_ARRAY_ELEMENT_TEXT 函數傳回 JSON 字串最外圍陣列中的 JSON 陣列元素 (採用以零開始的索引)。陣列的第一個元素在位置 0。如果索引是負數或超出邊界，JSON_EXTRACT_ARRAY_ELEMENT_TEXT 會傳回空字串。如果 null_if_invalid 引數設為 `true`，且 JSON 字串無效，此函數會傳回 NULL，而非傳回錯誤。

如需詳細資訊，請參閱 JSON 函數。

語法

```
JSON_EXTRACT_ARRAY_ELEMENT_TEXT('json string', pos [, null_if_invalid ] )
```

引數

json_string

格式正確的 JSON 字串。

pos

INTEGER，代表要傳回之陣列元素的索引 (採用以零開始的陣列索引)。

null_if_invalid

(選用) BOOLEAN 值，指定輸入 JSON 字串無效時是否傳回 NULL，而非傳回錯誤。若要在 JSON 無效時傳回 NULL，請指定 `true` (t)。若要在 JSON 無效時傳回錯誤，請指定 `false` (f)。預設值為 `false`。

傳回類型

VARCHAR

VARCHAR 字串，代表 pos 所參考的 JSON 陣列元素。

範例

若要傳回位置 2 的陣列元素 (即從零開始之陣列索引的第三個元素)，請使用下列範例。

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('[111,112,113]', 2);

+--------------------------------+
| json_extract_array_element_text |
+--------------------------------+
|                            113 |
+--------------------------------+
```

若要因 JSON 無效而傳回錯誤，請使用下列範例。

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('["a",["b",1,["c",2,3,null,]]]',1);

ERROR: invalid json array object ["a",["b",1,["c",2,3,null,]]]
```

若要將 null_if_invalid 設為 true，以便在 JSON 無效時，陳述式傳回 NULL 而非傳回錯誤，請使用下列範例。

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('["a",["b",1,["c",2,3,null,]]]',1,true);

+--------------------------------+
| json_extract_array_element_text |
+--------------------------------+
| NULL                           |
+--------------------------------+
```

JSON_EXTRACT_PATH_TEXT 函數

JSON_EXTRACT_PATH_TEXT 函數傳回 JSON 字串中由一連串路徑元素所參考的鍵-值對的值。JSON 路徑巢狀最深可達五層。路徑元素區分大小寫。如果 JSON 字串中沒有路徑元素，JSON_EXTRACT_PATH_TEXT 會傳回 NULL。

如果 null_if_invalid 引數設為 `true`，且 JSON 字串無效，此函數會傳回 NULL，而非傳回錯誤。

如需有關其他 JSON 函數的資訊，請參閱JSON 函數。如需使用 JSON 的相關資訊，請參閱從 JSON 格式 COPY。

語法

```
JSON_EXTRACT_PATH_TEXT('json_string', 'path_elem' [,'path_elem'[, …] ]
 [, null_if_invalid ] )
```

引數

json_string

格式正確的 JSON 字串。

path_elem

JSON 字串中的路徑元素。需要一個路徑元素。可指定其他路徑元素，最深可達五層。

null_if_invalid

(選用) BOOLEAN 值，指定輸入 JSON 字串無效時是否傳回 NULL，而非傳回錯誤。若要在 JSON 無效時傳回 NULL，請指定 `true` (t)。若要在 JSON 無效時傳回錯誤，請指定 `false` (f)。預設值為 `false`。

在 JSON 字串中，Amazon Redshift 將 \n 視為新行字元，將 \t 視為 Tab 字元。若要載入反斜線，請加上反斜線 (\\ ) 來逸出。如需詳細資訊，請參閱 JSON 中的逸出字元。

傳回類型

VARCHAR

VARCHAR 字串，代表路徑元素所參考的 JSON 值。

範例

若要傳回路徑 `'f4', 'f6'` 的值，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}','f4', 'f6');

+------------------------+
| json_extract_path_text |
+------------------------+
| star                   |
+------------------------+
```

若要因 JSON 無效而傳回錯誤，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4', 'f6');
```

```
ERROR: invalid json object {"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}
```

若要將 null_if_invalid 設為 true，以便在 JSON 無效時，陳述式傳回 NULL 而非傳回錯誤，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4',
 'f6',true);

+------------------------+
| json_extract_path_text |
+------------------------+
| NULL                   |
+------------------------+
```

若要傳回路徑 'farm', 'barn', 'color' 的值，其中擷取的值位於第三層，請使用下列範例。此範例使用 JSON lint 工具進行格式，以便於閱讀。

```
SELECT JSON_EXTRACT_PATH_TEXT('{
    "farm": {
        "barn": {
            "color": "red",
            "feed stocked": true
        }
    }
}', 'farm', 'barn', 'color');
+------------------------+
| json_extract_path_text |
+------------------------+
| red                    |
+------------------------+
```

若要在遺失 'color' 元素時傳回 NULL，請使用下列範例。此範例使用 JSON lint 工具進行格式化。

```
SELECT JSON_EXTRACT_PATH_TEXT('{
    "farm": {
        "barn": {}
    }
}', 'farm', 'barn', 'color');

+------------------------+
| json_extract_path_text |
+------------------------+
```

```
| NULL                    |
+-------------------------+
```

如果 JSON 有效，嘗試提取缺少的元素會傳回 NULL。

若要傳回路徑 'house', 'appliances', 'washing machine', 'brand' 的值，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{
  "house": {
    "address": {
      "street": "123 Any St.",
      "city": "Any Town",
      "state": "FL",
      "zip": "32830"
    },
    "bathroom": {
      "color": "green",
      "shower": true
    },
    "appliances": {
      "washing machine": {
        "brand": "Any Brand",
        "color": "beige"
      },
      "dryer": {
        "brand": "Any Brand",
        "color": "white"
      }
    }
  }
}', 'house', 'appliances', 'washing machine', 'brand');

+-------------------------+
| json_extract_path_text |
+-------------------------+
| Any Brand              |
+-------------------------+
```

下列範例會建立範例資料表，並以 SUPER 值填入資料表，然後傳回這兩個資料列'f2'的路徑值。

```
CREATE TABLE json_example(id INT, json_text SUPER);
```

```
INSERT INTO json_example VALUES
(1, JSON_PARSE('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}')),
(2, JSON_PARSE('{
    "farm": {
        "barn": {
            "color": "red",
            "feed stocked": true
        }
    }
}'));

SELECT * FROM json_example;
id          | json_text
------------+--------------------------------------------
1           | {"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}
2           | {"farm":{"barn":{"color":"red","feed stocked":true}}}


SELECT id, JSON_EXTRACT_PATH_TEXT(JSON_SERIALIZE(json_text), 'f2') FROM json_example;

id          | json_text
------------+--------------------------------------------
1           | {"f3":1}
2           |
```

JSON_PARSE 函數

JSON_PARSE 函數會剖析 JSON 格式的資料，並將其轉換為 SUPER 表示。

若要使用 INSERT 或 UPDATE 命令擷取到 SUPER 資料類型，請使用 JSON_PARSE 函數。當您使用 JSON_PARSE() 將 JSON 字串剖析為 SUPER 值時，適用某些限制。如需其他資訊，請參閱 SUPER 的剖析選項。

語法

```
JSON_PARSE( {json_string | binary_value} )
```

引數

json_string

以 VARBYTE 或 VARCHAR 類型傳回序列化 JSON 的運算式。

binary_value

VARBYTE 類型二進位值。

傳回類型

SUPER

範例

若要將 JSON 陣列 [10001,10002,"abc"] 轉換成 SUPER 資料類型，請使用下列範例。

```
SELECT JSON_PARSE('[10001,10002,"abc"]');

+---------------------+
|     json_parse      |
+---------------------+
| [10001,10002,"abc"] |
+---------------------+
```

若要確定函數將 JSON 陣列轉換成 SUPER 資料類型，請使用下列範例。如需更多資訊，請參閱JSON_TYPEOF 函數

```
SELECT JSON_TYPEOF(JSON_PARSE('[10001,10002,"abc"]'));

+-------------+
| json_typeof |
+-------------+
| array       |
+-------------+
```

CAN_JSON_PARSE 函數

CAN_JSON_PARSE 函數會剖析 JSON 格式的資料，如果可以使用 JSON_PARSE 函數將結果轉換為 SUPER 值，則傳回 true。

語法

```
CAN_JSON_PARSE( {json_string | binary_value} )
```

引數

json_string

以 VARBYTE 或 VARCHAR 形式傳回序列化 JSON 的運算式。

binary_value

VARBYTE 類型二進位值。

傳回類型

BOOLEAN

範例

若要查看 JSON 陣列 [10001,10002,"abc"] 是否可以轉換成 SUPER 資料類型，請使用下列範例。

```
SELECT CAN_JSON_PARSE('[10001,10002,"abc"]');

+-----------------+
| can_json_parse |
+-----------------+
| true           |
+-----------------+
```

JSON_SERIALIZE 函數

JSON_SERIALIZE 函數會根據 RFC 8259，將 SUPER 運算式序列化為文字 JSON 表示法。如需有關該 RFC 的詳細資訊，請參閱 JavaScript 物件標記法 (JSON) 資料交換格式。

SUPER 大小限制與區塊限制大致相同，且 VARCHAR 限制小於 SUPER 大小限制。因此，當 JSON 格式超過系統的 varchar 限制時，JSON_SERIALIZE 函數會傳回錯誤。若要檢查 SUPER 運算式的大小，請參閱 JSON_SIZE 函數。

語法

```
JSON_SERIALIZE(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

VARCHAR

範例

若要將 SUPER 值序列化為字串，請使用下列範例。

```
SELECT JSON_SERIALIZE(JSON_PARSE('[10001,10002,"abc"]'));

+---------------------+
|   json_serialize    |
+---------------------+
| [10001,10002,"abc"] |
+---------------------+
```

JSON_SERIALIZE_TO_VARBYTE 函數

JSON_SERIALIZE_TO_VARBYTE 函數會將 SUPER 值轉換為類似於 JSON_SERIALIZE()的 JSON 字串，但改為儲存在 VARBYTE 值中。

語法

```
JSON_SERIALIZE_TO_VARBYTE(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

VARBYTE

範例

若要序列化 SUPER 值，並以 VARBYTE 格式傳回結果，請使用下列範例。

```
SELECT JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]'));

+----------------------------------------+
|       json_serialize_to_varbyte        |
+----------------------------------------+
| 5b31303030312c31303030322c22616263225d |
+----------------------------------------+
```

若要序列化 SUPER 值並將結果轉換為 VARCHAR 格式，請使用下列範例。如需詳細資訊，請參閱 CAST 函數。

```
SELECT CAST((JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]'))) AS VARCHAR);

+---------------------------+
| json_serialize_to_varbyte |
+---------------------------+
| [10001,10002,"abc"]       |
+---------------------------+
```

機器學習函數

透過使用 Amazon Redshift 機器學習 (ML)，您可以使用 SQL 陳述式訓練 ML 模型，並在 SQL 查詢中調用它們以進行預測。Amazon Redshift 模型可解釋性包含特徵重要性值，可協助您了解訓練資料中的每個屬性如何影響預測結果。

接下來，您可以找到 Amazon Redshift 支援的 SQL 機器學習函數的說明。

主題

- EXPLAIN_MODEL 函數

EXPLAIN_MODEL 函數

EXPLAIN_MODEL 函數傳回 SUPER 資料類型，其中包含 JSON 格式的模型可解釋性報表。可解釋性報告包含有關所有模型特徵的 Shapley 值的資訊。

EXPLAIN_MODEL 函數目前僅支援 AUTO ON 或 AUTO OFF XGBoost 模型。

當可解釋性報告不可用時，函數會傳回顯示模型進度的狀態。其中包括 `Waiting for training job to complete`、`Waiting for processing job to complete` 和 `Processing job failed`。

當您執行 CREATE MODEL 陳述式時，解釋狀態會變成 `Waiting for training job to complete`。當模型經過訓練並傳送解釋請求時，解釋狀態會變成 `Waiting for processing job to complete`。當模型解釋成功完成後，即可獲得完整的可解釋性報告。否則狀態變成 `Processing job failed`。

當您執行 CREATE MODEL 陳述式時，您可以使用選用的 MAX_RUNTIME 參數來指定訓練應採取的時間上限。一旦模型建立達到這段時間，Amazon Redshift 就會停止建立模型。如果您在建立自動駕駛模型時達到該時間限制，Amazon Redshift 將傳回目前為止最佳模型。模型訓練完成後，模型可解釋性就可用，因此如果 MAX_RUNTIME 設定的時間較短，則可解釋性報告可能無法使用。訓練時間會因模型複雜度、資料大小及其他因素而有所不同。

語法

```
EXPLAIN_MODEL ('schema_name.model_name')
```

引數

schema_name

結構描述的名稱。如果未指定 schema_name，則會選取目前的結構描述。

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

傳回類型

EXPLAIN_MODEL 函數傳回 SUPER 資料類型，如下所示。

```
{"version":"1.0","explanations":{"kernel_shap":{"label0":{"global_shap_values":
{"x0":0.05,"x1":0.10,"x2":0.30,"x3":0.15},"expected_value":0.50}}}}
```

範例

下列範例會傳回解釋狀態 `waiting for training job to complete`。

```
select explain_model('customer_churn_auto_model');
```

```
                  explain_model
-----------------------------------------------------
{"explanations":"waiting for training job to complete"}
(1 row)
```

當模型解釋成功完成時，完整的可解釋性報告如下。

```
select explain_model('customer_churn_auto_model');
                                        explain_model
-----------------------------------------------------------------------------------------------
{"version":"1.0","explanations":{"kernel_shap":{"label0":{"global_shap_values":
{"x0":0.05386043365892927,"x1":0.10801289723274592,"x2":0.23227865827017378,"x3":0.067668513394
(1 row)
```

因為 EXPLAIIN_MODEL 函數會傳回 SUPER 資料類型，所以您可以查詢可解釋性報告。透過這樣做，您可以擷取 global_shap_values、expected_value 或特定於特徵的 Shapley 值。

以下範例會擷取模型的 global_shap_values。

```
select json_table.report.explanations.kernel_shap.label0.global_shap_values from
 (select explain_model('customer_churn_auto_model') as report) as json_table;
                                                    global_shap_values
-----------------------------------------------------------------------------------------------
{"state":0.10983770427197151,"account_length":0.1772441398408543,"area_code":0.0862682396863959
(1 row)
```

以下範例會擷取特徵 x0 的 global_shap_values。

```
select json_table.report.explanations.kernel_shap.label0.global_shap_values.x0 from
 (select explain_model('customer_churn_auto_model') as report) as json_table;
          x0
------------------------
  0.05386043365892927
(1 row)
```

如果模型是在一個特定的結構描述中建立的，且您有權存取建立的模型，那麼您可以查詢模型解釋，如下所示。

```
-- Check the current schema
SHOW search_path;
   search_path
-----------------
```

```
  $user, public
(1 row)
-- If you have the privilege to access the model explanation
-- in `test_schema`
SELECT explain_model('test_schema.test_model_name');
                        explain_model
-------------------------------------------------------------
{"explanations":"waiting for training job to complete"}
(1 row)
```

數學函數

主題

- 數學運算子符號
- ABS 函數
- ACOS 函數
- ASIN 函數
- ATAN 函數
- ATAN2 函數
- CBRT 函數
- CEILING (或 CEIL) 函數
- COS 函數
- COT 函數
- DEGREES 函數
- DEXP 函數
- DLOG1 函數
- DLOG10 函數
- EXP 函數
- FLOOR 函數
- LN 函數
- LOG 函數
- MOD 函數
- PI 函數
- POWER 函數

- RADIANS 函數
- RANDOM 函數
- ROUND 函數
- SIN 函數
- SIGN 函數
- SQRT 函數
- TAN 函數
- TRUNC 函數

本節描述 Amazon Redshift 中支援的數學運算子和函數。

數學運算子符號

下表列出支援的數學運算子。

支援的運算子

運算子	描述	範例	結果
+	加法	2 + 3	5
-	減法	2 - 3	-1
*	乘法	2 * 3	6
/	除法	4 /2	2
%	模數	5 % 4	1
^	指數	2.0 ^ 3.0	8
|/	平方根	| / 25.0	5
||/	立方根	|| / 27.0	3
@	絕對值	@ -5.0	5
<<	位元左移	1 << 4	16

運算子	描述	範例	結果
>>	位元右移	8 >> 2	2
&	位元 and	8 & 2	0

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要計算給定交易的已付佣金加 2.00 USD 手續費，請使用下列範例。

```
SELECT
    commission,
    (commission + 2.00) AS comm
FROM
    sales
WHERE
    salesid = 10000;

+------------+-------+
| commission | comm  |
+------------+-------+
|      28.05 | 30.05 |
+------------+-------+
```

若要計算給定交易售價的 20%，請使用下列範例。

```
SELECT pricepaid, (pricepaid * .20) as twentypct
FROM sales
WHERE salesid=10000;

+-----------+-----------+
| pricepaid | twentypct |
+-----------+-----------+
|       187 |      37.4 |
+-----------+-----------+
```

若要使用 DEXP 函數以根據持續成長模式來預測門票銷售，請使用下列範例。在此範例中，子查詢傳回 2008 年銷售的門票數。此結果以指數方式乘以過去 10 年的持續成長率 5%。

```
SELECT (SELECT SUM(qtysold) FROM sales, date
WHERE sales.dateid=date.dateid AND year=2008)^((5::float/100)*10) AS qty10years;

+------------------+
|    qty10years    |
+------------------+
| 587.664019657491 |
+------------------+
```

若要尋找日期 ID 大於或等於 2000 之銷售的支付總價和佣金，請使用下列範例。然後從支付總價中減去佣金總計。

```
SELECT SUM(pricepaid) AS sum_price, dateid,
SUM(commission) AS sum_comm, (SUM(pricepaid) - SUM(commission)) AS value
FROM sales
WHERE dateid >= 2000
GROUP BY dateid
ORDER BY dateid
LIMIT 10;

+-----------+--------+----------+-----------+
| sum_price | dateid | sum_comm |   value   |
+-----------+--------+----------+-----------+
|    305885 |   2000 | 45882.75 | 260002.25 |
|    316037 |   2001 | 47405.55 | 268631.45 |
|    358571 |   2002 | 53785.65 | 304785.35 |
|    366033 |   2003 | 54904.95 | 311128.05 |
|    307592 |   2004 |  46138.8 |  261453.2 |
|    333484 |   2005 |  50022.6 |  283461.4 |
|    317670 |   2006 |  47650.5 |  270019.5 |
|    351031 |   2007 | 52654.65 | 298376.35 |
|    313359 |   2008 | 47003.85 | 266355.15 |
|    323675 |   2009 | 48551.25 | 275123.75 |
+-----------+--------+----------+-----------+
```

ABS 函數

ABS 計算數字的絕對值，此數字可以是常值，或評估為數字的表達式。

語法

```
ABS(number)
```

引數

number

數字或評估為數字的表達式。它可以是 SMALLINT、INTEGER、BIGINT、DECIMAL、FLOAT4、FLOAT8 或 SUPER 類型。

傳回類型

ABS 傳回與其引數相同的資料類型。

範例

若要計算 -38 的絕對值，請使用下列範例。

```
SELECT ABS(-38);

+-----+
| abs |
+-----+
|  38 |
+-----+
```

若要計算 (14-76) 的絕對值，請使用下列範例。

```
SELECT ABS(14-76);

+-----+
| abs |
+-----+
|  62 |
+-----+
```

ACOS 函數

ACOS 是傳回數字反餘弦的三角函數。傳回值為弧度且介於 0 和 PI 之間。

語法

```
ACOS(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 -1 的反餘弦，請使用下列範例。

```
SELECT ACOS(-1);

+-------------------+
|       acos        |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

若要將 .5 的反餘弦轉換為同等度數，請使用下列範例。

```
SELECT (ACOS(.5) * 180/(SELECT PI())) AS degrees;

+-------------------+
|      degrees      |
+-------------------+
| 60.00000000000001 |
+-------------------+
```

ASIN 函數

ASIN 是傳回數字反正弦的三角函數。傳回值為弧度且介於 PI/2 和 -PI/2 之間。

語法

```
ASIN(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的正弦，請使用下列範例。

```
SELECT ASIN(1) AS halfpi;

+--------------------+
|       halfpi       |
+--------------------+
| 1.5707963267948966 |
+--------------------+
```

若要將 .5 的反正弦轉換為同等度數，請使用下列範例。

```
SELECT (ASIN(.5) * 180/(SELECT PI())) AS degrees;

+--------------------+
|      degrees       |
+--------------------+
| 30.000000000000004 |
+--------------------+
```

ATAN 函數

ATAN 是傳回數字反正切的三角函數。傳回值為弧度且介於 -PI 和 PI 之間。

語法

```
ATAN(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的反正切再乘以 4，請使用下列範例。

```
SELECT ATAN(1) * 4 AS pi;

+-------------------+
|        pi         |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

若要將 1 的反正切轉換為同等度數，請使用下列範例。

```
SELECT (ATAN(1) * 180/(SELECT PI())) AS degrees;

+---------+
| degrees |
+---------+
|      45 |
+---------+
```

ATAN2 函數

ATAN2 是傳回兩個數字相除之反正切的三角函數。傳回值為弧度且介於 PI/2 和 -PI/2 之間。

語法

```
ATAN2(number1, number2)
```

引數

number1

DOUBLE PRECISION 數字。

number2

DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的反正切再乘以 4，請使用下列範例。

```
SELECT ATAN2(2,2) * 4 AS PI;

+-------------------+
|        pi         |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

若要將 1 的的反正切 (計算結果為 0)轉換為同等度數，請使用下列範例。

```
SELECT (ATAN2(1,0) * 180/(SELECT PI())) AS degrees;

+---------+
| degrees |
+---------+
|      90 |
+---------+
```

CBRT 函數

CBRT 是計算指定數字立方根的數學函數。

語法

```
CBRT(number)
```

引數

CBRT 需要一個 DOUBLE PRECISION 數字作為引數。

傳回類型

DOUBLE PRECISION

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要計算給定交易之已付佣金的立方根，請使用下列範例。

```
SELECT CBRT(commission) FROM sales WHERE salesid=10000;

+--------------------+
|        cbrt        |
+--------------------+
| 3.0383953904884344 |
+--------------------+
```

CEILING (或 CEIL) 函數

CEILING 或 CEIL 函數用來將數字捨進到下一個整數。(FLOOR 函數 將數字捨去到下一個整數。)

語法

```
{CEIL | CEILING}(number)
```

引數

number

數字或評估為數字的運算式。它可以是 SMALLINT、INTEGER、BIGINT、DECIMAL、FLOAT4、FLOAT8 或 SUPER 類型。

傳回類型

CEILING 和 CEIL 傳回相同的資料類型作為它的引數。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 null。

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要計算給定銷售交易之已付佣金的上限，請使用下列範例。

```
SELECT CEILING(commission) FROM sales
WHERE salesid=10000;

+---------+
| ceiling |
+---------+
|      29 |
+---------+
```

COS 函數

COS 是傳回數字餘弦的三角函數。傳回值為弧度且介於 -1 和 1 (含) 之間。

語法

```
COS(double_precision)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

COS 函數會傳回 DOUBLE PRECISION 數字。

範例

若要傳回 0 的餘弦，請使用下列範例。

```
SELECT COS(0);
```

```
+-----+
| cos |
+-----+
|   1 |
+-----+
```

若要傳回 pi 的餘弦，請使用下列範例。

```
SELECT COS(PI());

+-----+
| cos |
+-----+
|  -1 |
+-----+
```

COT 函數

COT 是傳回數字餘切的三角函數。輸入參數必須不是零。

語法

```
COT(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的餘切，請使用下列範例。

```
SELECT COT(1);

+--------------------+
|        cot         |
```

```
+--------------------+
| 0.6420926159343306 |
+--------------------+
```

DEGREES 函數

將角度的弧度轉換為其同等度數。

語法

```
DEGREES(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 0.5 弧度的同等度數，請使用下列範例。

```
SELECT DEGREES(.5);

+-------------------+
|      degrees      |
+-------------------+
| 28.64788975654116 |
+-------------------+
```

若要將 PI 弧度轉換為度數，請使用下列範例。

```
SELECT DEGREES(pi());

+---------+
| degrees |
+---------+
|     180 |
```

```
+---------+
```

DEXP 函數

DEXP 函數以科學記號表示法傳回雙精確度數字的指數值。DEXP 和 EXP 函數的唯一差異是 DEXP 的參數必須為 DOUBLE PRECISION。

語法

```
DEXP(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

使用 DEXP 函數以根據持續成長模式來預測門票銷售。在此範例中，子查詢傳回 2008 年銷售的門票數。此結果乘以 DEXP 函數的結果，而此函數指出過去 10 年的持續成長率為 7%。

```
SELECT (SELECT SUM(qtysold)
FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * DEXP((7::FLOAT/100)*10) qty2010;

+-------------------+
|      qty2010      |
+-------------------+
| 695447.4837722216 |
+-------------------+
```

DLOG1 函數

DLOG1 函數傳回輸入參數的自然對數。LN 函數 的同義詞。

DLOG10 函數

DLOG10 傳回輸入參數的以 10 為底的對數。

LOG 函數 的同義詞。

語法

```
DLOG10(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回數字 100 的以 10 為底的對數，請使用下列範例。

```
SELECT DLOG10(100);

+--------+
| dlog10 |
+--------+
|      2 |
+--------+
```

EXP 函數

EXP 函數會實作數值表達式的指數函數，或是自然對數的底數，表達式的 e 次方。EXP 函數為 LN 函數 的倒數。

語法

```
EXP(expression)
```

引數

運算式

運算式必須是 INTEGER、DECIMAL 或 DOUBLE PRECISION 資料類型。

傳回類型

DOUBLE PRECISION

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

使用 EXP 函數以根據持續成長模式來預測門票銷售。在此範例中，子查詢傳回 2008 年銷售的門票數。此結果乘以 EXP 函數的結果，而此函數指出過去 10 年的持續成長率為 7%。

```
SELECT (SELECT SUM(qtysold)
FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * EXP((7::FLOAT/100)*10) qty2018;

+-------------------+
|      qty2018      |
+-------------------+
| 695447.4837722216 |
+-------------------+
```

FLOOR 函數

FLOOR 函數將數字捨去到下一個整數。

語法

```
FLOOR(number)
```

引數

number

數字或評估為數字的運算式。它可以是 SMALLINT、INTEGER、BIGINT、DECIMAL、FLOAT4、FLOAT8 或 SUPER 類型。

傳回類型

FLOOR 傳回與其引數相同的資料類型。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要顯示使用 FLOOD 函數之前和之後，針對指定的銷售交易支付之佣金的數值，請使用下列範例。

```
SELECT commission
FROM sales
WHERE salesid=10000;

+------------+
| commission |
+------------+
|      28.05 |
+------------+

SELECT FLOOR(commission)
FROM sales
WHERE salesid=10000;

+-------+
| floor |
+-------+
|    28 |
+-------+
```

LN 函數

傳回輸入參數的自然對數。

DLOG1 函數 的同義詞。

語法

```
LN(expression)
```

引數

運算式

函數運算的目標欄或表達式。

> **Note**
>
> 如果運算式參考 Amazon Redshift 使用者建立的資料表或 Amazon Redshift STL 或 STV 系統資料表，此函數會針對某些資料類型傳回錯誤。

如果具有下列資料類型的表達式參考使用者建立的資料表或系統資料表，則會產生錯誤。具有這些資料類型的表達式只能在領導者節點上執行：

- BOOLEAN
- CHAR
- DATE
- DECIMAL 或 NUMERIC
- TIMESTAMP
- VARCHAR

在使用者建立的資料表和 STL 或 STV 系統資料表上，具有下列資料類型的表達式可以成功執行：

- BIGINT
- DOUBLE PRECISION
- INTEGER
- REAL
- SMALLINT

傳回類型

LN 函數傳回與輸入運算式相同的類型。

範例

若要傳回數字 2.718281828 的自然對數或以 e 為底的對數，請使用下列範例。

```
SELECT LN(2.718281828);

+--------------------+
|         ln         |
+--------------------+
| 0.9999999998311267 |
+--------------------+
```

請注意，答案幾乎等於 1。

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要傳回 USERS 資料表的 userid 欄中值的自然對數，請使用下列範例。

```
SELECT username, LN(userid) FROM users ORDER BY userid LIMIT 10;

+----------+--------------------+
| username |         ln         |
+----------+--------------------+
| JSG99FHE |                  0 |
| PGL08LJI | 0.6931471805599453 |
| IFT66TXU | 1.0986122886681098 |
| XDZ38RDD | 1.3862943611198906 |
| AEB55QTM | 1.6094379124341003 |
| NDQ15VBM |  1.791759469228055 |
| OWY35QYB | 1.9459101490553132 |
| AZG78YIP | 2.0794415416798357 |
| MSD36KVR | 2.1972245773362196 |
| WKW41AIW |  2.302585092994046 |
+----------+--------------------+
```

LOG 函數

傳回數字的對數。

如果您使用此函數計算以 10 為底的對數，則也可以使用 DLOG10 函數。

語法

```
LOG([base, ]argument)
```

參數

base

(選用) 對數函數的底。這個數字必須是正數且不能等於 1。如果省略此參數，Amazon Redshift 會計算 argument 的以 10 為底的對數。

argument

對數函數的引數。此數字必須為正數。如果 argument 值是 1，則函數傳回 0。

傳回類型

LOG 函數會傳回 DOUBLE PRECISION 數字。

範例

若要尋找 100 的以 2 為底的對數，請使用下列範例。

```
SELECT LOG(2, 100);
+-------------------+
|        log        |
+-------------------+
| 6.643856189774725 |
+-------------------+
```

若要尋找 100 的以 10 為底的對數，請使用下列範例。請注意，如果您省略基本參數，Amazon Redshift 假設基數為 10。

```
SELECT LOG(100);

+-----+
| log |
+-----+
|   2 |
+-----+
```

MOD 函數

傳回兩個數字的餘數，也稱為模數運算。為了計算結果，第一個參數除以第二個參數。

語法

```
MOD(number1, number2)
```

引數

number1

第一個輸入參數是 INTEGER、SMALLINT、BIGINT 或 DECIMAL 數字。如果任一參數為 DECIMAL 類型，則另一個參數也必須為 DECIMAL 類型。如果任一參數為 INTEGER，則另一個參數可以是 INTEGER、SMALLINT 或 BIGINT。兩個參數也都可以是 SMALLINT 或 BIGINT，但如果一個參數是 BIGINT，則另一個參數不能是 SMALLINT。

number2

第二個參數是 INTEGER、SMALLINT、BIGINT 或 DECIMAL 數字。相同的資料類型規則適用於 number2 與 number1。

傳回類型

如果兩個輸入參數都是相同類型，則 MOD 函數的傳回類型與輸入參數的數值類型相同。不過，如果任一輸入參數為 INTEGER，則傳回類型也會是 INTEGER。有效的傳回類型為 DECIMAL、INT、SMALLINT 和 BIGINT。

使用須知

您可以使用 % 作為模數運算子。

範例

若要傳回一個數字除以另一個數字時的餘數，請使用下列範例。

```
SELECT MOD(10, 4);

+-----+
| mod |
+-----+
|   2 |
+-----+
```

若要在使用 MOD 函數時傳回 DECIMAL 結果，請使用下列範例。

```
SELECT MOD(10.5, 4);

+-----+
| mod |
+-----+
| 2.5 |
+-----+
```

若要在執行 MOD 函數之前轉換數字，請使用下列範例。如需詳細資訊，請參閱 CAST 函數。

```
SELECT MOD(CAST(16.4 AS INTEGER), 5);

+-----+
| mod |
+-----+
|   1 |
+-----+
```

若要透過除以 2 來檢查第一個參數是否為偶數，請使用下列範例。

```
SELECT mod(5,2) = 0 AS is_even;

+---------+
| is_even |
+---------+
| false   |
+---------+
```

若要使用 % 做為模數運算子，請使用下列範例。

```
SELECT 11 % 4 as remainder;

 +-----------+
| remainder |
+-----------+
|         3 |
+-----------+
```

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要傳回 CATEGORY 資料表中奇數編號類別的資訊，請使用下列範例。

```
SELECT catid, catname
FROM category
WHERE MOD(catid,2)=1
ORDER BY 1,2;

+-------+-----------+
| catid |  catname  |
+-------+-----------+
|     1 | MLB       |
|     3 | NFL       |
|     5 | MLS       |
|     7 | Plays     |
|     9 | Pop       |
|    11 | Classical |
+-------+-----------+
```

PI 函數

PI 函數會傳回 pi 的值，精確到 14 位小數。

語法

```
PI()
```

傳回類型

DOUBLE PRECISION

範例

若要傳回 pi 的值，請使用下列範例。

```
SELECT PI();

+------------------+
|        pi        |
+------------------+
| 3.141592653589793 |
+------------------+
```

POWER 函數

POWER 函數是將一個數值表達式乘上以第二個數值表達式為次方的指數函數。例如，2 的三次方以 POWER(2,3) 計算，結果為 8。

語法

```
{POW | POWER}(expression1, expression2)
```

引數

expression1

要乘以次方的數值表達式。必須是 INTEGER、DECIMAL 或 FLOAT 資料類型。

expression2

expression1 要乘以的次方。必須是 INTEGER、DECIMAL 或 FLOAT 資料類型。

傳回類型

DOUBLE PRECISION

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

在下列範例中，使用 POWER 函數以根據 2008 年銷售的門票數 (子查詢的結果)，預測未來 10 年的門票銷售。在此範例中，成長率設為每年 7%。

```
SELECT (SELECT SUM(qtysold) FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * POW((1+7::FLOAT/100),10) qty2010;

+-------------------+
|      qty2010      |
+-------------------+
| 679353.7540885945 |
+-------------------+
```

下列範例是上一個範例的變化版，成長率每年 7%，但間隔設為月份 (10 年共 120 個月)。

```
SELECT (SELECT SUM(qtysold) FROM sales, date
```

```
WHERE sales.dateid=date.dateid
AND year=2008) * POW((1+7::FLOAT/100/12),120) qty2010;

+-----------------+
|     qty2010     |
+-----------------+
| 694034.54678046 |
+-----------------+
```

RADIANS 函數

RADIANS 函數會將以度為單位的角度轉換為其同等弧度。

語法

```
RADIANS(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 180 度的同等弧度，請使用下列範例。

```
SELECT RADIANS(180);

+-------------------+
|      radians      |
+-------------------+
| 3.141592653589793 |
+-------------------+
```

RANDOM 函數

RANDOM 函數會產生介於 0.0 (包含) 到 1.0 (不包含) 間的隨機值。

語法

```
RANDOM()
```

傳回類型

DOUBLE PRECISION

使用須知

以 SET 命令設定種子值之後呼叫 RANDOM，讓 RANDOM 以可預測的序列產生數字。

範例

若要計算 0 到 99 之間的隨機值，請使用下列範例。如果隨機數字為 0 到 1，此查詢會產生 0 到 100 的隨機數字。

```
SELECT CAST(RANDOM() * 100 AS INT);

+------+
| int4 |
+------+
|   59 |
+------+
```

此範例會使用 SET 命令設定 SEED 值，讓 RANDOM 產生可預測的數字序列。

若要在不設定 SEED 值的情況下傳回三個 RANDOM 整數，請使用下列範例。

```
SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|    6 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   68 |
+------+
```

```
SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   56 |
+------+
```

若要將 SEED 值設定為 .25，並傳回三個以上的亂數，請使用下列範例。

```
SET SEED TO .25;
SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   21 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   79 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   12 |
+------+
```

若要將 SEED 值重設為 .25，並驗證 RANDOM 是否傳回與前三個呼叫相同的結果，請使用下列範例。

```
SET SEED TO .25;
SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   21 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
```

```
+------+
| int4 |
+------+
|   79 |
+------+

SELECT CAST(RANDOM() * 100 AS INT);
+------+
| int4 |
+------+
|   12 |
+------+
```

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要從 SALES 表格擷取 10 個項目的統一隨機樣本，請使用下列範例。

```
SELECT *
FROM sales
ORDER BY RANDOM()
LIMIT 10;

+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|   45422 |  51114 |     5983 |   24482 |    4369 |   2118 |       1 |       195 |
 29.25 | 2008-10-19 05:20:07 |
|   42481 |  47638 |     4573 |    6198 |    6479 |   1987 |       4 |      1140 |
   171 | 2008-06-10 09:39:19 |
|   31494 |  34759 |    18895 |    4719 |    7753 |   2090 |       4 |      1024 |
 153.6 | 2008-09-21 03:44:26 |
|  119388 | 136685 |    21815 |   41905 |    2071 |   1884 |       1 |       359 |
 53.85 | 2008-02-27 10:43:10 |
|  166990 | 225037 |    18529 |    7628 |     746 |   2113 |       1 |      2009 |
 301.35 | 2008-10-14 10:07:44 |
|   11146 |  12096 |    42685 |    6619 |    1876 |   2123 |       1 |        29 |
  4.35 | 2008-10-24 06:23:54 |
|  148537 | 172056 |    15102 |   11787 |    6122 |   1923 |       2 |       480 |
    72 | 2008-04-07 03:58:23 |
|   68945 |  78387 |     7359 |   18323 |    6636 |   1910 |       1 |       457 |
 68.55 | 2008-03-25 08:31:03 |
```

```
|   52796 |  59576 |     9909 |   15102 |    7958 |   1951 |       1 |       479 |
 71.85 | 2008-05-05 02:25:08 |
|   90684 | 103522 |    38052 |   21549 |    7384 |   2117 |       1 |       313 |
 46.95 | 2008-10-18 05:43:11 |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

若要擷取 10 個商品的隨機樣本，但請依價格比例來選擇商品，請使用下列範例。例如，一個商品的價格如果是其他商品的兩倍，則出現在查詢結果的機率也是其他商品的兩倍。

```
SELECT *
FROM sales
ORDER BY -LOG(RANDOM()) / pricepaid
LIMIT 10;

+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
 commission |      saletime       |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
|  158340 | 208208 |    17082 |   42018 |    1211 |   2160 |       4 |      6852 |
 1027.8 | 2008-11-30 12:21:43 |
|   53250 |  60069 |    12644 |    7066 |    7942 |   1838 |       4 |      1528 |
 229.2 | 2008-01-12 11:24:56 |
|   22929 |  24938 |    47314 |    6503 |     179 |   2000 |       3 |       741 |
 111.15 | 2008-06-23 08:04:50 |
|  164980 | 221181 |     1949 |   19670 |    1471 |   1906 |       1 |      1330 |
 199.5 | 2008-03-21 07:59:51 |
|  159641 | 211179 |    44897 |   16652 |    7458 |   2128 |       1 |      1019 |
 152.85 | 2008-10-29 02:02:15 |
|   73143 |  83439 |     5716 |    5727 |    7314 |   1903 |       1 |       248 |
  37.2 | 2008-03-18 11:07:42 |
|   84778 |  96749 |    46608 |   32980 |    3883 |   1999 |       2 |       958 |
 143.7 | 2008-06-22 12:13:31 |
|  171096 | 232929 |    43683 |    8536 |    8353 |   1870 |       1 |       929 |
 139.35 | 2008-02-13 01:36:36 |
|   74212 |  84697 |    39809 |   15569 |    5525 |   2105 |       2 |       896 |
 134.4 | 2008-10-06 11:47:50 |
|  158011 | 207556 |    25399 |   16881 |     232 |   2088 |       2 |      2526 |
 378.9 | 2008-09-19 06:00:26 |
+---------+--------+----------+---------+---------+--------+---------+-----------
+------------+---------------------+
```

ROUND 函數

ROUND 函數將數字四捨五入至最接近的整數或小數。

ROUND 函數可以選擇包含第二個引數做為 INTEGER，表示四捨五入的小數位數 (任一方向)。當您未提供第二個引數時，函數會四捨五入為最接近的整數。指定第二個引數 integer 時，函數會捨入為具有 integer 小數位數的最接近的數字。

語法

```
ROUND(number [ , integer ] )
```

引數

number

數字或評估為數字的運算式。它可以是 DECIMAL、FLOAT8 或 SUPER 類型。Amazon Redshift 可以隱含轉換其他數字資料類型。

integer

(選用) INTEGER，表示沿任一方向捨入的小數位數。此引數不支援 SUPER 資料類型。

傳回類型

ROUND 會傳回與輸入 number 相同的數值資料類型。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要將給定交易的已付佣金四捨五入至最接近的整數，請使用下列範例。

```
SELECT commission, ROUND(commission)
FROM sales WHERE salesid=10000;

+------------+-------+
| commission | round |
+------------+-------+
|      28.05 |    28 |
```

```
+------------+-------+
```

若要將給定交易的已付佣金四捨五入至第一位小數，請使用下列範例。

```
SELECT commission, ROUND(commission, 1)
FROM sales WHERE salesid=10000;

+------------+-------+
| commission | round |
+------------+-------+
|      28.05 |  28.1 |
+------------+-------+
```

若要以上一個範例的反方向延伸精確度，請使用下列範例。

```
SELECT commission, ROUND(commission, -1)
FROM sales WHERE salesid=10000;

+------------+-------+
| commission | round |
+------------+-------+
|      28.05 |    30 |
+------------+-------+
```

SIN 函數

SIN 是傳回數字正弦的三角函數。傳回值介於 -1 和 1 之間。

語法

```
SIN(number)
```

引數

number

以弧度表示的 DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 -PI 的正弦，請使用下列範例。

```
SELECT SIN(-PI());

+-------------------------+
|           sin           |
+-------------------------+
| -0.00000000000000012246 |
+-------------------------+
```

SIGN 函數

SIGN 函數傳回數字的符號 (正或負)。如果引數是正的，SIGN 函數的結果會是 1，如果引數是負的則結果為 -1，如果引數是 0 則結果為 0。

語法

```
SIGN(number)
```

引數

number

數字或評估為數字的表達式。它可以是 DECIMAL、FLOAT8 或 SUPER 類型。Amazon Redshift 可以根據隱含轉換規則轉換其他資料類型。

傳回類型

SIGN 傳回與輸入引數相同的數值資料類型。如果輸入是 DECIMAL，則輸出為 DECIMAL(1,0)。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列範例顯示表格 t2 中的 d 欄具有 DOUBLE PRECISION 類型，因為輸入是 DOUBLE PRECISION，而表格 t2 中的 n 欄具有 NUMERIC(1,0) 輸出，因為輸入是 NUMERIC。

```
CREATE TABLE t1(d DOUBLE PRECISION, n NUMERIC(12, 2));
```

```
INSERT INTO t1 VALUES (4.25, 4.25), (-4.25, -4.25);
CREATE TABLE t2 AS SELECT SIGN(d) AS d, SIGN(n) AS n FROM t1;
SELECT table_name, column_name, data_type FROM SVV_REDSHIFT_COLUMNS WHERE
 table_name='t1' OR table_name='t2';

+------------+-------------+----------------------+
| table_name | column_name |       data_type      |
+------------+-------------+----------------------+
| t1         | d           | double precision     |
| t1         | n           | numeric(12,2)        |
| t2         | d           | double precision     |
| t2         | n           | numeric(1,0)         |
| t1         | col1        | character varying(20) |
+------------+-------------+----------------------+
```

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要確定從 SALES 表中為給定交易支付的佣金的符號，請使用以下範例。

```
SELECT commission, SIGN(commission)
FROM sales WHERE salesid=10000;

+------------+------+
| commission | sign |
+------------+------+
|      28.05 |    1 |
+------------+------+
```

SQRT 函數

SQRT 函數傳回 NUMERIC 值的平方根。平方根是一個數字與其自身相乘以獲得給定值。

語法

```
SQRT(expression)
```

引數

運算式

運算式必須具有 INTEGER、DECIMAL 或 FLOAT 資料類型，或隱含轉換為這些資料類型的資料類型。expression 可以包含函數。

傳回類型

DOUBLE PRECISION

範例

若要傳回 16 的平方根，請使用下列範例。

```
SELECT SQRT(16);

+------+
| sqrt |
+------+
|    4 |
+------+
```

若要使用隱含類型轉換傳回字串 16 的平方根，請使用下列範例。

```
SELECT SQRT('16');

+------+
| sqrt |
+------+
|    4 |
+------+
```

若要在使用 ROUND 函數之後傳回 16.4 的平方根，請使用下列範例。

```
SELECT SQRT(ROUND(16.4));

+------+
| sqrt |
+------+
|    4 |
+------+
```

若要在指定圓的面積時傳回半徑的長度，請使用下列範例。例如，當以平方英吋為單位指定面積時，它會以英吋為單位計算半徑。樣本中的面積為 20。

```
SELECT SQRT(20/PI()) AS radius;

+--------------------+
```

```
|      radius        |
+--------------------+
| 2.5231325220201604 |
+--------------------+
```

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要從 SALES 資料表中傳回某些 COMMISSION 值的平方根，請使用下列範例。COMMISSION 欄是 DECIMAL 欄。此範例顯示如何在具有更複雜條件式邏輯的查詢中使用函數。

```
SELECT SQRT(commission)
FROM sales WHERE salesid < 10 ORDER BY salesid;

+--------------------+
|        sqrt        |
+--------------------+
| 10.449880382090505 |
| 3.3763886032268267 |
|  7.245688373094719 |
|  5.123475382979799 |
|  4.806245936279167 |
|  7.687652437513028 |
| 10.871982339941507 |
| 5.4359911699707535 |
|   9.41541289588513 |
+--------------------+
```

若要傳回同一組 COMMISSION 值的四捨五入平方根，請使用下列範例。

```
SELECT ROUND(SQRT(commission))
FROM sales WHERE salesid < 10 ORDER BY salesid;

+-------+
| round |
+-------+
|    10 |
|     3 |
|     7 |
|     5 |
|     5 |
|     8 |
|    11 |
|     5 |
```

```
|     9 |
+-------+
```

TAN 函數

TAN 是傳回數字正切的三角函數。輸入引數是一個數字 (以弧度表示)。

語法

```
TAN(number)
```

引數

number

DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 0 的正切，請使用下列範例。

```
SELECT TAN(0);

+-----+
| tan |
+-----+
|   0 |
+-----+
```

TRUNC 函數

TRUNC 函數將數字截斷為先前的整數或小數。

TRUNC 函數可以選擇包含第二個引數做為 INTEGER，表示四捨五入的小數位數 (任一方向)。當您未提供第二個引數時，函數會四捨五入為最接近的整數。指定第二個引數 integer 時，函數會捨入為具有 integer 小數位數的最接近的數字。

這個函數也可以截斷 TIMESTAMP 並傳回 DATE。如需詳細資訊，請參閱 TRUNC 函數。

語法

```
TRUNC(number [ , integer ])
```

引數

number

數字或評估為數字的運算式。它可以是 DECIMAL、FLOAT8 或 SUPER 類型。Amazon Redshift 可以根據隱含轉換規則轉換其他資料類型。

integer

(選用) INTEGER，表示精確度的小數位數 (任一方向)。如果未提供整數，數字會截斷為整數；如果指定整數，數字會截斷至指定的小數位數。不支援 SUPER 資料類型。

傳回類型

TRUNC 會傳回與輸入 number 相同的資料類型。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列部分範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

若要截斷給定銷售交易的已付佣金，請使用下列範例。

```
SELECT commission, TRUNC(commission)
FROM sales WHERE salesid=784;

+------------+-------+
| commission | trunc |
+------------+-------+
|     111.15 |   111 |
+------------+-------+
```

若要將同一個佣金值截斷至第一位小數，請使用下列範例。

```
SELECT commission, TRUNC(commission,1)
```

```
FROM sales WHERE salesid=784;

+------------+-------+
| commission | trunc |
+------------+-------+
|     111.15 | 111.1 |
+------------+-------+
```

若要使用第二個引數的負值截斷佣金，請使用下列範例。請注意 111.15 向下捨入為 110。

```
SELECT commission, TRUNC(commission,-1)
FROM sales WHERE salesid=784;

+------------+-------+
| commission | trunc |
+------------+-------+
|     111.15 |   110 |
+------------+-------+
```

物件函數

以下是 Amazon Redshift 支援建立 SUPER 類型物件的 SQL 物件函數：

主題

- 下限屬性名稱函數
- OBJECT 函數
- OBJECT_TRANSFORM 函數
- 上限屬性名稱函數

下限屬性名稱函數

將 SUPER 值中所有適用的屬性名稱轉換為小寫，並使用與LOWER 函數. 名稱支援 UTF-8 多位元組字元，每個字元最多四個位元組。

若要將 SUPER 屬性名稱轉換為大寫，請使用上限屬性名稱函數。

語法

```
LOWER_ATTRIBUTE_NAMES(super_expression)
```

引數

super_expression

一個超級表達式。

傳回類型

SUPER

使用須知

在 Amazon Redshift 中，欄標識符傳統上不區分大小寫，並轉換為小寫。如果您從 JSON 等區分大小寫的資料格式擷取資料，則資料可能包含大小寫混合的屬性名稱。

請考量下列範例。

```
CREATE TABLE t1 (s) AS SELECT JSON_PARSE('{"AttributeName": "Value"}');


SELECT s.AttributeName FROM t1;

attributename
-------------
NULL


SELECT s."AttributeName" FROM t1;

attributename
-------------
NULL
```

Amazon Redshift 對於這兩個查詢都返回 NULL。若要查詢AttributeName，請使用 DELUMBER_NAME 將資料的屬性名稱轉換為小寫。請考量下列範例。

```
CREATE TABLE t2 (s) AS SELECT LOWER_ATTRIBUTE_NAMES(s) FROM t1;


SELECT s.attributename FROM t2;

attributename
```

```
-------------
"Value"


SELECT s.AttributeName FROM t2;

attributename
-------------
"Value"


SELECT s."attributename" FROM t2;

attributename
-------------
"Value"


SELECT s."AttributeName" FROM t2;

attributename
-------------
"Value"
```

使用混合大小寫物件屬性名稱的相關選項是enable_case_sensitive_super_attribute組態選項，可讓 Amazon Redshift 辨識 SUPER 屬性名稱中的大小寫。這可以是使用下拉屬性名稱的替代解決方案。如需相關資訊enable_case_sensitive_super_attribute，請移至enable_case_sensitive_super_attribute。

範例

將 SUPER 屬性名稱轉換為小寫

下列範例會使用 DELUMER 屬性名稱來轉換資料表中所有 SUPER 值的屬性名稱。

```
-- Create a table and insert several SUPER values.
CREATE TABLE t (i INT, s SUPER);

INSERT INTO t VALUES
  (1, NULL),
  (2, 'A'::SUPER),
  (3, JSON_PARSE('{"AttributeName": "B"}')),
  (4, JSON_PARSE(
```

```
     '[{"Subobject": {"C": "C"},
        "Subarray": [{"D": "D"}, "E"]
      }]'));

-- Convert all attribute names to lowercase.
UPDATE t SET s = LOWER_ATTRIBUTE_NAMES(s);

SELECT i, s FROM t ORDER BY i;

 i |                         s
---+---------------------------------------------------
 1 | NULL
 2 | "A"
 3 | {"attributename":"B"}
 4 | [{"subobject":{"c":"C"},"subarray":[{"d":"D"}, "E"]}]
```

觀察如何降低屬性名稱的功能。

- NULL 值和標量超級值，如"A"不變。
- 在 SUPER 物件中，所有屬性名稱都變更為小寫，但屬性值 (例如) "B" 保持不變。
- 降低屬性 _NAMES 遞歸地應用於嵌套在超級數組內或另一個對象內的任何超級對象。

在具有重複屬性名稱的超級對象上使用降低屬性名稱

如果一個超級對象包含的屬性，其名稱僅在他們的大小寫不同，降低屬性 _NAMES 將引發一個錯誤。請考量下列範例。

```
SELECT LOWER_ATTRIBUTE_NAMES(JSON_PARSE('{"A": "A", "a": "a"}'));

error:   Invalid input
code:    8001
context: SUPER value has duplicate attributes after case conversion.
```

OBJECT 函數

建立 SUPER 資料類型的物件。

語法

```
OBJECT ( [ key1, value1 ], [ key2, value2 ...] )
```

引數

key1, key2

計算結果為 VARCHAR 類型字串的運算式。

value1, value2

除了日期時間類型以外的任何 Amazon Redshift 資料類型的運算式，因為 Amazon Redshift 不會將日期時間類型轉換為 SUPER 資料類型。如需日期時間類型的相關資訊，請參閱日期時間 (Datetime) 類型。

物件中的 value 運算式不需要是相同的資料類型。

旋轉類型

SUPER

範例

```
-- Creates an empty object.
select object();

object
--------
{}
(1 row)

-- Creates objects with different keys and values.
select object('a', 1, 'b', true, 'c', 3.14);

object
---------------------------
{"a":1,"b":true,"c":3.14}
(1 row)

select object('a', object('aa', 1), 'b', array(2,3), 'c', json_parse('{}'));

object
---------------------------------
{"a":{"aa":1},"b":[2,3],"c":{}}
(1 row)
```

```
-- Creates objects using columns from a table.
create table bar (k varchar, v super);
insert into bar values ('k1', json_parse('[1]')), ('k2', json_parse('{}'));
select object(k, v) from bar;

object
------------
{"k1":[1]}
{"k2":{}}
(2 rows)

-- Errors out because DATE type values can't be converted to SUPER type.
select object('k', '2008-12-31'::date);

ERROR:  OBJECT could not convert type date to super
```

OBJECT_TRANSFORM 函數

變換 SUPER 物件。

語法

```
OBJECT_TRANSFORM(
  input
  [KEEP path1, ...]
  [SET
    path1, value1,
    ...,  ...
  ]
)
```

引數

input

解析為 SUPER 類型物件的表達式。

KEEP

在這個子句中指定的所有路徑值都會保留，並轉移到輸出物件。

此子句是選用的。

path1、path2...

常數字串常值，採用以句點分隔的雙引號路徑元件的格式。例如，'"a"."b"."c"' 是有效的路徑值。這適用於 KEEP 和 SET 子句中的路徑參數。

SET

path 和 value 配對可修改現有路徑或新增路徑，並在輸出物件中設定該路徑的值。

此子句是選用的。

value1、value2...

解析為 SUPER 類型值的表達式。請注意，數字、文字和布林值類型可以解析為 SUPER。

傳回類型

SUPER

使用須知

OBJECT_TRANSFORM 會傳回 SUPER 類型物件，其中包含來自 KEEP 中指定的 input路徑值，以及 SET 中指定的 path 和 value 對。

如果 KEEP 和 SET 都是空的，OBJECT_TRANSFORM 會傳回 input。

如果 input 不是 SUPER 類型 object，則 OBJECT_TRANSFORM 會傳回 input，而不管任何 KEEP 或 SET 值。

範例

下列範例會將 SUPER 物件轉換成另一個 SUPER 物件。

```
CREATE TABLE employees (
    col_person SUPER
);

INSERT INTO employees
VALUES
    (
        json_parse('
            {
                "name": {
                    "first": "John",
                    "last": "Doe"
```

```
                },
                "age": 25,
                "ssn": "111-22-3333",
                "company": "Company Inc.",
                "country": "U.S."
            }
        ')
    ),
    (
        json_parse('
            {
                "name": {
                    "first": "Jane",
                    "last": "Appleseed"
                },
                "age": 34,
                "ssn": "444-55-7777",
                "company": "Organization Org.",
                "country": "Ukraine"
            }
        ')
    )
;

SELECT
    OBJECT_TRANSFORM(
        col_person
        KEEP
            '"name"."first"',
            '"age"',
            '"company"',
            '"country"'
        SET
            '"name"."first"', UPPER(col_person.name.first::TEXT),
            '"age"', col_person.age + 5,
            '"company"', 'Amazon'
    ) AS col_person_transformed
FROM employees;

--This result is formatted for ease of reading.
                  col_person_transformed
-------------------------------------------------------------
{
    "name": {
```

```
        "first": "JOHN"
    },
    "age": 30,
    "company": "Amazon",
    "country": "U.S."
}
{
    "name": {
        "first": "JANE"
    },
    "age": 39,
    "company": "Amazon",
    "country": "Ukraine"
}
```

上限屬性名稱函數

將 SUPER 值中所有適用的屬性名稱轉換為大寫，並使用與UPPER 函數. 名稱支援 UTF-8 多位元組字元，每個字元最多四個位元組。

若要將 SUPER 屬性名稱轉換為小寫，請使用下限屬性名稱函數。

語法

```
UPPER_ATTRIBUTE_NAMES(super_expression)
```

引數

super_expression

一個超級表達式。

傳回類型

SUPER

範例

將 SUPER 屬性名稱轉換為大寫

下列範例會使用 UPPER_Attribute_NAME 來轉換資料表中所有 SUPER 值的屬性名稱。

```
-- Create a table and insert several SUPER values.
CREATE TABLE t (i INT, s SUPER);

INSERT INTO t VALUES
  (1, NULL),
  (2, 'a'::SUPER),
  (3, JSON_PARSE('{"AttributeName": "b"}')),
  (4, JSON_PARSE(
     '[{"Subobject": {"c": "c"},
       "Subarray": [{"d": "d"}, "e"]
      }]'));

-- Convert all attribute names to uppercase.
UPDATE t SET s = UPPER_ATTRIBUTE_NAMES(s);

SELECT i, s FROM t ORDER BY i;

 i |                         s
---+---------------------------------------------------
 1 | NULL
 2 | "a"
 3 | {"ATTRIBUTENAME":"B"}
 4 | [{"SUBOBJECT":{"C":"c"},"SUBARRAY":[{"D":"d"}, "e"]}]
```

觀察屬性名稱的功能。

- NULL 值和標量超級值，如"a"不變。
- 在 SUPER 物件中，所有屬性名稱都變更為大寫，但屬性值 (如"b"保持不變)。
- UPPER_Attribute_NAMES 遞歸地套用至巢狀在超級陣列內或其他物件內的任何超級物件。

在具有重複屬性名稱的超級對象上使用 UPPER_Attribute_NAME

如果超級對象包含的屬性，其名稱僅在他們的大小寫不同，UPPER_AUTITE_NAMES 將引發一個錯誤。請考量下列範例。

```
SELECT UPPER_ATTRIBUTE_NAMES(JSON_PARSE('{"A": "A", "a": "a"}'));

error:   Invalid input
code:    8001
context: SUPER value has duplicate attributes after case conversion.
```

空間函數

幾何物件之間的關係是以維度延伸九交模型 (DE-9IM) 為基礎。這個模型定義了如相等、包含和涵蓋等述詞。如需空間關係定義的相關資訊，請參閱 Wikipedia 中的 DE-9IM。

如需如何將空間資料與 Amazon Redshift 搭配使用的相關資訊，請參閱在 Amazon Redshift 中查詢空間資料。

Amazon Redshift 提供了可與 GEOMETRY 和 GEOGRAPHY 資料類型一起使用的空間函數。以下列出支援 GEOGRAPHY 資料類型的函數：

- ST_Area
- ST_AsEWKT
- JSON AsGeo
- ST _ EWKB AsHex
- ST_ 白板 AsHex
- ST_ AsText
- ST_Distance
- ST_ GeogFromText
- ST_ 白板 GeogFrom
- ST_Length
- ST_NPoints
- ST_Perimeter

以下列出 Amazon Redshift 支援的完整空間函數集。

主題

- AddBBox
- DropBBox
- GeometryType
- H3_ FromLongLat
- H3_ FromPoint
- H3_Polyfill
- ST_ AddPoint

- ST_Angle
- ST_Area
- ST_ AsBinary
- ST_AsEWKB
- ST_AsEWKT
- JSON AsGeo
- ST_ 白板 AsHex
- ST _ EWKB AsHex
- ST_ AsText
- ST_Azimuth
- ST_Boundary
- ST_Buffer
- ST_Centroid
- ST_Collect
- ST_Contains
- ST_ ContainsProperly
- ST_ ConvexHull
- ST_ CoveredBy
- ST_Covers
- ST_Crosses
- ST_Dimension
- ST_Disjoint
- ST_Distance
- ST_ DistanceSphere
- ST_DWithin
- ST_ EndPoint
- ST_Envelope
- ST_Equals
- ST_ ExteriorRing
- ST_Force2D

- ST_Force3D
- ST_Force3DM
- ST_Force3DZ
- ST_Force4D
- ST_ GeoHash
- ST_ GeogFromText
- ST_ 白板 GeogFrom
- ST_GeometryN
- ST_ GeometryType
- ST _ EWKB GeomFrom
- ST-尤克特 GeomFrom
- ST_ GeomFromGeoHash
- JSON GeomFromGeo
- ST_ GeomFromGeoSquare
- ST_ GeomFromText
- ST_ 白板 GeomFrom
- ST_ GeoSquare
- ST_ N InteriorRing
- ST_Intersects
- ST_Intersection
- ST_ 逆時鐘 IsPolygon
- ST_ 密西西比 IsPolygon
- ST_ IsClosed
- ST_ IsCollection
- ST_ IsEmpty
- ST_ IsRing
- ST_ IsSimple
- ST_ IsValid
- ST_Length
- ST_ LengthSphere

- ST_Length2D
- ST_ LineFromMultiPoint
- ST_ LineInterpolatePoint
- ST_M
- ST_ MakeEnvelope
- ST_ MakeLine
- ST_ MakePoint
- ST_ MakePolygon
- ST_ MemSize
- ST_MMax
- ST_MMin
- ST_Multi
- ST_NDims
- ST_NPoints
- ST_NRings
- ST_ NumGeometries
- ST_ NumInteriorRings
- ST_ NumPoints
- ST_Perimeter
- ST_Perimeter2D
- ST_Point
- ST_PointN
- ST_Points
- ST_Polygon
- ST_ RemovePoint
- ST_Reverse
- ST_ SetPoint
- ST_SetSRID
- ST_Simplify
- ST_SRID

- ST_ StartPoint
- ST_Touches
- ST_Transform
- ST_Union
- ST_Within
- ST_X
- ST_XMax
- ST_XMin
- ST_Y
- ST_YMax
- ST_YMin
- ST_Z
- ST_ZMax
- ST_ZMin
- SupportsBBox

AddBBox

AdDbBox 會傳回支援使用預先計算的週框方塊編碼輸入幾何的副本。如需週框方塊支援的相關資訊，請參閱邊界框。

語法

```
AddBBox(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會傳回支援使用週框方塊編碼的輸入多邊形幾何的副本。

```
SELECT ST_AsText(AddBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
 st_astext
----------
 POLYGON((0 0,1 0,0 1,0 0))
```

DropBBox

DropbBox 會傳回不支援使用預先計算的邊界方框編碼的輸入幾何副本。如需週框方塊支援的相關資訊，請參閱邊界框。

語法

```
DropBBox(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回不支援使用週框方塊編碼之輸入多邊形幾何的副本。

```
SELECT ST_AsText(DropBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
 st_astext
----------
```

```
POLYGON((0 0,1 0,0 1,0 0))
```

GeometryType

GeometryType 以字串形式傳回輸入幾何的子類型。

語法

```
GeometryType(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

VARCHAR 表示 geom 的子類型。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回 2D、3DZ、4D 幾何的字串值	傳回 3DM 幾何的字串值	幾何子類型
POINT	POINTM	在 geom 是 POINT 子類型時傳回
LINESTRING	LINESTRINGM	在 geom 是 LINESTRING 子類型時傳回
POLYGON	POLYGONM	在 geom 是 POLYGON 子類型時傳回
MULTIPOINT	MULTIPOINTM	在 geom 是 MULTIPOINT 子類型時傳回
MULTILINESTRING	MULTILINESTRINGM	在 geom 是 MULTILINESTRING 子類型時傳回

傳回 2D、3DZ、4D 幾何的字串值	傳回 3DM 幾何的字串值	幾何子類型
MULTIPOLYGON	MULTIPOLYGONM	在 geom 是 MULTIPOLYGON 子類型時傳回
GEOMETRYCOLLECTION	GEOMETRYCOLLECTIONM	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會轉換多邊形的已知文字 (WKT) 表示法，並將 GEOMETRY 子類型做為字串傳回。

```
SELECT GeometryType(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
geometrytype
-------------
 POLYGON
```

H3_ FromLongLat

H3_ 從輸入經度、緯度和解析度FromLongLat 傳回對應的 H3 儲存格 ID。如需 H3 索引的詳細資訊，請參閱 H3。

語法

```
H3_FromLongLat(longitude, lattitude, resolution)
```

引數

經度

DOUBLE PRECISION 資料類型的值，或是評估為 DOUBLE PRECISION 類型的表達式。

緯度

DOUBLE PRECISION 資料類型的值，或是評估為 DOUBLE PRECISION 類型的表達式。

解析度

INTEGER 資料類型的值，或是評估為 INTEGER 類型的表達式。該值表示 H3 網格系統的解析度。該值必須是介於 0 和 15 之間的整數。0 是最普遍的，15 是最好。

傳回類型

BIGINT— 代表 H3 儲存格 ID。

如果解析度超出範圍，則會傳回錯誤。

範例

下列 SQL 會從經度 0、緯度 0 和解析度 10 傳回 H3 儲存格 ID。

```
SELECT H3_FromLongLat(0, 0, 10);
```

```
 h3_fromlonglat
-------------------
 623560421467684863
```

H3_ FromPoint

H3_ 從輸入幾何點和解析度FromPoint 傳回對應的 H3 零件 ID。如需 H3 索引的詳細資訊，請參閱 H3。

語法

```
H3_FromPoint(geom, resolution)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。geom 必須是 POINT。

解析度

INTEGER 資料類型的值，或是評估為 INTEGER 類型的表達式。該值表示 H3 網格系統的解析度。該值必須是介於 0 和 15 之間的整數。0 是最普遍的，15 是最好。

傳回類型

BIGINT— 代表 H3 儲存格 ID。

如果 geom 不是 POINT，則會傳回錯誤。

如果解析度超出範圍，則會傳回錯誤。

如果 geom 為空白，則會傳回 NULL。

範例

下方的 SQL 從點 0,0 和解析度 10 傳回 H3 儲存格 ID。

```
SELECT H3_FromPoint(ST_GeomFromText('POINT(0 0)'), 10);
```

```
 h3_frompoint
--------------------
 623560421467684863
```

H3_Polyfill

H3_Polyfill 會傳回對應於指定解析度之輸入多邊形中包含的六邊形和五邊形的對應 H3 儲存格 ID。如需 H3 索引的詳細資訊，請參閱 H3。

語法

```
H3_Polyfill(geom, resolution)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。geom 必須是 POLYGON。

解析度

INTEGER 資料類型的值，或是評估為 INTEGER 類型的表達式。該值表示 H3 網格系統的解析度。該值必須是介於 0 和 15 之間的整數。0 是最普遍的，15 是最好。

傳回類型

SUPER— 代表 H3 儲存格 ID 的清單。

如果 geom 不是 POLYGON，則會傳回錯誤。

如果解析度超出範圍，則會傳回錯誤。

如果 geom 為空白，則會傳回 NULL。

範例

下方的 SQL 會從多邊形和解析度 4 傳回 H3 儲存格 ID 的 SUPER 資料類型陣列。

```
SELECT H3_Polyfill(ST_GeomFromText('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))'), 4);
```

```
 h3_polyfill
-------------------------------------------------------------------------------------------------
 [596538848238895103,596538805289222143,596538856828829695,596538813879156735,5965379205259591
```

ST_ AddPoint

ST_ 會AddPoint 傳回與加入點的輸入幾何圖形相同的線串幾何圖形。如果提供索引，則點會新增在索引位置。如果索引為 -1 或未提供，則點附加至 linestring。

索引是以零開始。結果的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

傳回幾何的維度與 geom1 值的維度相同。如果 geom1 和 geom2 具有不同的維度，則將 geom2 投影到 geom1 的維度。

語法

```
ST_AddPoint(geom1, geom2)
```

```
ST_AddPoint(geom1, geom2, index)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。該點可以是空點。

索引

資料類型 INTEGER 的值，代表以零開始的索引位置。

傳回類型

GEOMETRY

如果 geom1、geom2 或 index 為 Null，則會傳回 Null。

如果 geom2 是空點，則傳回 geom1 的副本。

如果 geom1 不是 LINESTRING，則會傳回錯誤。

如果 geom2 不是 POINT，則會傳回錯誤。

如果 index 超出範圍，則會傳回錯誤。索引位置的有效值是 -1 或是介於 0 與 ST_NumPoints(geom1) 之間的值。

範例

下列 SQL 會將一點新增到 linestring，使其成為一個封閉的 linestring。

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_AddPoint(g, ST_StartPoint(g))) FROM tmp;
```

```
 st_asewkt
-------------------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 5,0 5,0 0)
```

下列 SQL 會將一點新增到 linestring 中的特定位置。

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_AddPoint(g, ST_SetSRID(ST_Point(5, 10), 4326), 3)) FROM tmp;
```

```
 st_asewkt
-----------------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 10,5 5,0 5)
```

ST_Angle

ST_Angle 傳回以弧度為單位順時針測量的點之間的角度，如下所示：

- 如果輸入三個點，則將測量傳回的角度 P1-P2-P3，就像繞 P2 順時針從 P1 旋轉到 P3 獲得的角度一樣。
- 如果輸入四個點，則傳回由有向線 P1-P2 和 P3-P4 形成的順時針角度。如果輸入是退化情況 (也就是說，P1 等於 P2，或 P3 等於 P4)，則傳回 null。

傳回值以弧度為單位，範圍為 [0, 2π)。

ST_Angle 對輸入幾何的 2D 投影進行操作。

語法

```
ST_Angle(geom1, geom2, geom3)
```

```
ST_Angle(geom1, geom2, geom3, geom4)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

geom3

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

geom4

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

傳回類型

DOUBLE PRECISION.

如果 geom1 等於 geom2，或 geom2 等於 geom3，則傳回 null。

如果 geom1、geom2、geom3 或 geom4 為 null，則傳回 null。

如果任何 geom1、geom2、geom3 或 geom4 是空點，則傳回錯誤。

如果 geom1、geom2、geom3 和 geom4 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

範例

下列 SQL 會傳回三個輸入點轉換為度數的角度。

```
SELECT ST_Angle(ST_Point(1,1), ST_Point(0,0), ST_Point(1,0)) / Pi() * 180.0 AS angle;
```

```
 angle
---------------
    45
```

下列 SQL 會傳回四個輸入點轉換為度數的角度。

```
SELECT ST_Angle(ST_Point(1,1), ST_Point(0,0), ST_Point(1,0), ST_Point(2,0)) / Pi() *
 180.0 AS angle;
```

```
 angle
---------------
   225
```

ST_Area

對於輸入幾何，ST_Area 傳回 2D 投影的笛卡爾面積。面積單位與表示輸入幾何座標的單位相同。對於點、linestring、multipoint 和 multilinestring，函數會傳回 0。對於幾何集合，它會傳回集合中幾何面積的總和。

對於輸入地理，ST_Area 會傳回在球體 (由 SRID 決定) 上所計算輸入面積地理之 2D 投影的測地線面積。長度單位為平方公尺。對於點、multipoint 和線形地理，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中面積地理的面積總和。

語法

```
ST_Area(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

DOUBLE PRECISION

如果 geo 為 null，則傳回 null。

範例

下列 SQL 會傳回 multipolygon 的笛卡爾面積。

```
SELECT ST_Area(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20 10,10
 0)))'));
```

```
 st_area
---------
     100
```

以下 SQL 傳回地理中多邊形的面積。

```
SELECT ST_Area(ST_GeogFromText('polygon((34 35, 28 30, 25 34, 34 35))'));
```

```
     st_area
------------------
 201824655743.383
```

下列 SQL 針對線性地理傳回零。

```
SELECT ST_Area(ST_GeogFromText('multipoint(0 0, 1 1, -21.32 121.2)'));
```

```
 st_area
---------
       0
```

ST_ AsBinary

ST_ 會AsBinary 傳回輸入幾何的十六進位已知二進位 (WKB) 表示法。對於 3DZ、3DM 和 4D 幾何圖形，ST_ AsBinary 使用開放地理空間聯盟 (OGC) 標準值作為幾何類型。

語法

```
ST_AsBinary(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

VARBYTE

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回多邊形的十六進位 WKB 表示法。

```
SELECT ST_AsBinary(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_asbinary
--------------------------------
0103000000010000000500000000000000000000000000000000000000000000000000F03F00000
```

ST_AsEWKB

ST_AsEWKB 會傳回輸入幾何的擴充已知二進位 (EWKB) 表示法。對於 3DZ、3DM 和 4D 幾何，ST_AsEWKB 使用開放地理空間聯盟 (OGC) 標準值作為幾何類型。

語法

```
ST_AsEWKB(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

VARBYTE

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回多邊形的十六進位 EWKB 表示法。

```
SELECT ST_AsEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_asewkb
---------------------------------
0103000020E6100000010000000500000000000000000000000000000000000000000000000000F03F000000000000F03F000000000000F03F000000000000F03F0000000000000000000000000000000000000000000000000
```

ST_AsEWKT

ST_AsEWKT 會傳回輸入幾何或地理的擴充已知文字 (EWKT) 表示法。對於 3DZ、3DM 和 4D 幾何，ST_AsEWKT 會將 Z、M 或 ZM 附加至幾何類型的 WKT 值。

語法

```
ST_AsEWKT(geo)
```

```
ST_AsEWKT(geo, precision)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

precision

INTEGER 資料類型的值。對於幾何，使用指定的精確度 1-20 顯示 geo 座標。如果沒有指定 precision，則預設為 15。對於地理，使用指定的精確度顯示 geo 座標。如果沒有指定 precision，則預設為 15。

傳回類型

VARCHAR

如果 geo 為 null，則傳回 null。

如果 precision 為 Null，則會傳回 Null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回 linestring 的 EWKT 表示法。

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326));
```

```
st_asewkt
---------------------------------
 SRID=4326;LINESTRING(3.14159265358979 -6.28318530717959,2.71828182845905
 -1.41421356237309)
```

以下 SQL 會傳回 linestring 的 EWKT 表示法。幾何的座標會使用六位數的精確度顯示。

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326), 6);
```

```
st_asewkt
-------------------------------
 SRID=4326;LINESTRING(3.14159 -6.28319,2.71828 -1.41421)
```

以下 SQL 會傳回地理的 EWKT 表示法。

```
SELECT ST_AsEWKT(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
                  st_asewkt
---------------------------------------------
 SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

JSON AsGeo

AsGeoJSON 會傳回輸入幾何或地理位置的 GeoJSON 表示法。如需 GeoJSON 的相關資訊，請參閱 Wikipedia 中的 GeoJSON。

對於 3DZ 和 4D 幾何，輸出幾何是輸入 3DZ 或 4D 幾何的 3DZ 投影。也就是說，x、y 和 z 座標存在於輸出中。對於 3DM 幾何，輸出幾何是輸入 3DM 幾何的 2D 投影。也就是說，只有 x 和 y 座標存在於輸出中。

對於輸入的地理位置，ST_ AsGeo JSON 會傳回輸入地理位置的 GeoJSON 表示法。使用指定的精確度顯示地理座標。

語法

```
ST_AsGeoJSON(geo)
```

```
ST_AsGeoJSON(geo, precision)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

precision

INTEGER 資料類型的值。對於幾何，使用指定的精確度 1-20 顯示 geo 座標。如果沒有指定 precision，則預設為 15。對於地理，使用指定的精確度顯示 geo 座標。如果沒有指定 precision，則預設為 15。

傳回類型

VARCHAR

如果 geo 為 null，則傳回 null。

如果 precision 為 Null，則會傳回 Null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回 linestring 的 GeoJSON 表示法。

```
SELECT ST_AsGeoJSON(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)'));
```

```
st_asgeojson
--------------------------------------------------------------------------------------------------
 {"type":"LineString","coordinates":[[3.14159265358979,-6.28318530717959],
[2.71828182845905,-1.4142135623709]]}
```

以下 SQL 會傳回 linestring 的 GeoJSON 表示法。幾何的座標會使用六位數的精確度顯示。

```
SELECT ST_AsGeoJSON(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)'), 6);
```

```
st_asgeojson
-----------------------------------------------------------------------------
  {"type":"LineString","coordinates":[[3.14159,-6.28319],[2.71828,-1.41421]]}
```

以下 SQL 會傳回地理的 GeoJSON 表示法。

```
SELECT ST_AsGeoJSON(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
                             st_asgeojson
------------------------------------------------------------------------
 {"type":"LineString","coordinates":[[110,40],[2,3],[-10,80],[-7,9]]}
```

ST_ 白板 AsHex

ST_ AsHex WKB 會使用 ASCII 十六進位字元 (0—9、A — F) 傳回輸入幾何或地理位置的十六進位已知二進位 (WKB) 表示。對於 3DZ、3DM 和 4D 幾何圖形或地理位置，ST_ AsHex WKB 會針對幾何圖形或地理類型使用開放地理空間協會 (OGC) 標準值。

語法

```
ST_AsHexWKB(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

VARCHAR

如果 geo 為 null，則傳回 null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回幾何中多邊形的十六進位 WKB 表示法。

```
SELECT ST_AsHexWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_ashexwkb
```

```
--------------------------------
0103000000010000000500000000000000000000000000000000000000000000000000F03F00000
```

以下 SQL 會傳回地理中多邊形的十六進位 WKB 表示法。

```
SELECT ST_AsHexWKB(ST_GeogFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'));
```

```
st_ashexwkb
--------------------------------
0103000000010000000500000000000000000000000000000000000000000000000000F03F00000
```

ST _ EWKB AsHex

ST_ AsHex EWKB 會傳回使用 ASCII 十六進位字元 (0-9、A — F) 輸入幾何或地理位置的延伸已知二進位 (EWKB) 表示法。對於 3DZ、3DM 和 4D 幾何圖形或地理位置，ST_ AsHex EWKB 會針對幾何圖形或地理類型使用 PostGIS 延伸的 WKB 值。

語法

```
ST_AsHexEWKB(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

VARCHAR

如果 geo 為 null，則傳回 null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回幾何中多邊形的十六進位 EWKB 表示法。

```
SELECT ST_AsHexEWKB(ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326));
```

```
st_ashexewkb
--------------------------------
0103000020E6100000010000000500000000000000000000000000000000000000000000000000F03F000000000000F03F000000000000F03F000000000000F03F0000000000000000000000000000000000
```

以下 SQL 會傳回地理中多邊形的十六進位 EWKB 表示法。

```
SELECT ST_AsHexEWKB(ST_GeogFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))'));
```

```
st_ashexewkb
--------------------------------
0103000020E6100000010000000500000000000000000000000000000000000000000000000000F03F000000000000F03F000000000000F03F000000000000F03F0000000000000000000000000000000000
```

ST_ AsText

ST_ 會AsText 傳回輸入幾何圖形或地理的已知文字 (WKT) 表示法。對於 3DZ、3DM 和 4D 幾何或地理，ST_AsEWKT 會將 Z、M 或 ZM 附加至幾何或地理類型的 WKT 值。

語法

```
ST_AsText(geo)
```

```
ST_AsText(geo, precision)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

precision

INTEGER 資料類型的值。對於幾何，使用指定的精確度 1-20 顯示 geo 座標。如果沒有指定 precision，則預設為 15。對於地理，使用指定的精確度顯示 geo 座標。如果沒有指定 precision，則預設為 15。

傳回類型

VARCHAR

如果 geo 為 null，則傳回 null。

如果 precision 為 Null，則會傳回 Null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回 linestring 的 WKT 表示法。

```
SELECT ST_AsText(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326));
```

```
st_astext
-------------------------------
LINESTRING(3.14159265358979 -6.28318530717959,2.71828182845905 -1.41421356237309)
```

以下 SQL 會傳回 linestring 的 WKT 表示法。幾何的座標會使用六位數的精確度顯示。

```
SELECT ST_AsText(ST_GeomFromText('LINESTRING(3.141592653589793
 -6.283185307179586,2.718281828459045 -1.414213562373095)', 4326), 6);
```

```
st_astext
---------------------------------------------
 LINESTRING(3.14159 -6.28319,2.71828 -1.41421)
```

以下 SQL 會傳回地理的 WKT 表示法。

```
SELECT ST_AsText(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
             st_astext
-----------------------------------
 LINESTRING(110 40,2 3,-10 80,-7 9)
```

ST_Azimuth

ST_Azimuth 會使用兩個輸入點的 2D 投影傳回以北為基礎的笛卡爾方位。

語法

```
ST_Azimuth(point1, point2)
```

引數

point1

GEOMETRY 資料類型的 POINT 值。point1 的空間參考系統識別碼 (SRID) 必須與 point2 的 SRID 相符。

point2

GEOMETRY 資料類型的 POINT 值。point2 的 SRID 必須與 point1 的 SRID 相符。

傳回類型

數字，為 DOUBLE PRECISION 資料類型的角度 (弧度)。值的範圍介於 0 (含) 到 2 pi (不含)。

如果 point1 或 point2 是空點，則會傳回錯誤。

如果 point1 或 point2 為 Null，則會傳回 Null。

如果 point1 和 point2 相等，則會傳回 Null。

如果 point1 或 point2 不是點，則會傳回錯誤。

如果 point1 和 point2 沒有空間參考系統識別碼 (SRID) 的值，則會傳回錯誤。

範例

以下 SQL 會傳回輸入點的方位。

```
SELECT ST_Azimuth(ST_Point(1,2), ST_Point(5,6));
```

```
st_azimuth
-------------------
```

```
0.7853981633974483
```

ST_Boundary

ST_Boundary 會傳回輸入幾何的邊界，如下所示：

- 如果輸入幾何是空的 (也就是說，它不包含任何點)，則會按原樣傳回。
- 如果輸入幾何是點或非空的多點，則會傳回空的幾何集合。
- 如果輸入為 linestring 或 multilinestring，則會傳回包含邊界上所有點的多點。多點可能是空的)。
- 如果輸入是沒有任何內環的多邊形，則會傳回表示其邊界的封閉 linestring。
- 如果輸入是具有內環的多邊形，或者是多重多邊形，則會傳回 multilinestring。Multilinestring 包含面積幾何中所有環的所有邊界，做為封閉 linestring。

為了確定點相等，ST_Boundary 對輸入幾何的 2D 投影進行操作。如果輸入幾何為空，則會以與輸入相同的維度傳回其副本。對於非空的 3DM 和 4D 幾何，它們的 m 座標會被捨棄。在 3DZ 和 4D multilinestring 的特殊情況下，multilinestring 邊界點的 z 座標會計算為具有相同 2D 投影之 linestring 邊界點之相異 z 值的平均值。

語法

```
ST_Boundary(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 GEOMETRYCOLLECTION，則會傳回錯誤。

範例

下列 SQL 傳回輸入多邊形的邊界作為 multilinestring。

```
SELECT ST_AsEWKT(ST_Boundary(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1
 2,2 1,1 1))')));
```

```
st_asewkt
--------------------
 MULTILINESTRING((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 1,1 1))
```

ST_Buffer

ST_Buffer 傳回 2D 幾何，該幾何表示與投影在 xy 笛卡爾平面上的輸入幾何的距離小於或等於輸入距離的所有點。

語法

```
ST_Buffer(geom, distance)
```

```
ST_Buffer(geom, distance, number_of_segments_per_quarter_circle)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

distance

DOUBLE PRECISION 資料類型的值，表示緩衝區的距離 (或半徑)。

number_of_segments_per_quarter_circle

INTEGER 資料類型的值。此值決定了圍繞輸入幾何的每個頂點近似四分之一圓的點數。負值預設為零。預設值為 8。

傳回類型

GEOMETRY

ST_Buffer 函數會傳回 xy 笛卡爾平面中的二維 (2D) 幾何。

如果 geom 是 GEOMETRYCOLLECTION，則會傳回錯誤。

範例

以下 SQL 會傳回輸入 linestring 的緩衝區。

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('LINESTRING(1 2,5 2,5 8)'), 2));
```

```
               st_asewkt
POLYGON((-1 2,-0.96157056080646 2.39018064403226,-0.847759065022573
 2.76536686473018,-0.662939224605089 3.11114046603921,-0.414213562373093
 3.4142135623731,-0.111140466039201 3.66293922460509,0.234633135269824
 3.84775906502257,0.609819355967748 3.96157056080646,1 4,3 4,3 8,3.03842943919354
 8.39018064403226,3.15224093497743 8.76536686473018,3.33706077539491
 9.11114046603921,3.58578643762691 9.4142135623731,3.8888595339608
 9.66293922460509,4.23463313526982 9.84775906502257,4.60981935596775
 9.96157056080646,5 10,5.39018064403226 9.96157056080646,5.76536686473018
 9.84775906502257,6.11114046603921 9.66293922460509,6.4142135623731
 9.41421356237309,6.66293922460509 9.1111404660392,6.84775906502258
 8.76536686473017,6.96157056080646 8.39018064403225,7 8,7 2,6.96157056080646
 1.60981935596774,6.84775906502257 1.23463313526982,6.66293922460509
 0.888859533960796,6.41421356237309 0.585786437626905,6.1111404660392
 0.33706077539491,5.76536686473018 0.152240934977427,5.39018064403226
 0.0384294391935391,5 0,1 0,0.609819355967744 0.0384294391935391,0.234633135269821
 0.152240934977427,-0.111140466039204 0.337060775394909,-0.414213562373095
 0.585786437626905,-0.662939224605091 0.888859533960796,-0.847759065022574
 1.23463313526982,-0.961570560806461 1.60981935596774,-1 2))
```

以下 SQL 會傳回近似於圓的輸入點幾何圖形的緩衝區。由於此命令未指定每四分之一圓的線段數，因此函數使用八個線段的預設值來近似四分之一圓。

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('POINT(3 4)'), 2));
```

```
               st_asewkt
POLYGON((1 4,1.03842943919354 4.39018064403226,1.15224093497743
 4.76536686473018,1.33706077539491 5.11114046603921,1.58578643762691
 5.4142135623731,1.8888595339608 5.66293922460509,2.23463313526982
 5.84775906502257,2.60981935596775 5.96157056080646,3 6,3.39018064403226
 5.96157056080646,3.76536686473019 5.84775906502257,4.11114046603921
 5.66293922460509,4.4142135623731 5.41421356237309,4.66293922460509
 5.1111404660392,4.84775906502258 4.76536686473017,4.96157056080646 4.39018064403225,5
 4,4.96157056080646 3.60981935596774,4.84775906502257 3.23463313526982,4.66293922460509
 2.8888595339608,4.41421356237309 2.58578643762691,4.1111404660392
 2.33706077539491,3.76536686473018 2.15224093497743,3.39018064403226 2.03842943919354,3
```

```
 2,2.60981935596774 2.03842943919354,2.23463313526982 2.15224093497743,1.8888595339608
 2.33706077539491,1.58578643762691 2.58578643762691,1.33706077539491
 2.8888595339608,1.15224093497743 3.23463313526982,1.03842943919354 3.60981935596774,1
 4))
```

以下 SQL 會傳回近似於圓的輸入點幾何圖形的緩衝區。由於此命令指定 3 作為每四分之一圓的線段數，因此函數使用三個線段來近似四分之一圓。

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('POINT(3 4)'), 2, 3));
```

```
               st_asewkt
POLYGON((1 4,1.26794919243112 5,2 5.73205080756888,3 6,4
 5.73205080756888,4.73205080756888 5,5 4,4.73205080756888 3,4 2.26794919243112,3 2,2
 2.26794919243112,1.26794919243112 3,1 4))
```

ST_Centroid

st_Centroid 會傳回表示幾何質心的點，如下所示：

- 對於 POINT 幾何，它會傳回座標為幾何中點座標平均值的點。
- 對於 LINESTRING 幾何，它會傳回一個點，其座標是幾何體各段中點的加權平均值，其中權重是幾何體各段的長度。
- 對於 POLYGON 幾何，它會傳回一個點，其座標是面積幾何三角剖分質心的加權平均值，其中權重是三角剖分中三角形的面積。
- 對於幾何集合，它會傳回幾何集合中最大拓撲維度的幾何質心的加權平均值。

語法

```
ST_Centroid(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空白，則會傳回 Null。

範例

以下 SQL 會傳回輸入 linestring 的中心點。

```
SELECT ST_AsEWKT(ST_Centroid(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9, -22
 -33)', 4326)))
```

```
                     st_asewkt
-----------------------------------------------------
 SRID=4326;POINT(15.6965103455214 27.0206782881905)
```

ST_Collect

ST_Collect 有兩個變體。一個接受兩個幾何，一個接受彙總運算式。

ST_Collect 的第一個變體會從輸入幾何建立幾何。會保留輸入幾何的順序。此變體的運作方式如下：

- 如果兩個輸入幾何都是點，則傳回具有兩個點的 MULTIPOINT。
- 如果兩個輸入幾何都是 linestring，則傳回具有兩個 linestring 的 MULTILINESTRING。
- 如果兩個輸入幾何都是多邊形，則傳回具有兩個多邊形的 MULTIPOLYGON。
- 否則，會傳回具有兩個輸入幾何的 GEOMETRYCOLLECTION。

ST_Collect 的第二個變體會根據幾何欄中的幾何建立幾何。幾何形狀沒有確定的傳回順序。指定 WITHIN GROUP (ORDER BY ...) 子句以指定傳回幾何的順序。此變體的運作方式如下：

- 如果輸入彙總運算式中的所有非 NULL 列都是點，則傳回包含彙總運算式中所有點的多點。
- 如果彙總運算式中的所有非 NULL 列都是 linestring，則會傳回包含彙總運算式中所有 linestring 的 multilinestring。
- 如果彙總運算式中的所有非 NULL 列都是多邊形，則結果是傳回一個包含彙總運算式中所有多邊形的多重多邊形。
- 否則，會傳回包含彙總運算式中所有幾何的 GEOMETRYCOLLECTION。

ST_Collect 會傳回與輸入幾何相同維度的幾何。所有輸入幾何必須具有相同的維度。

語法

```
ST_Collect(geom1, geom2)
```

```
ST_Collect(aggregate_expression)  [WITHIN GROUP (ORDER BY sort_expression1 [ASC | DESC]
 [, sort_expression2 [ASC | DESC] ...])]
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

aggregate_expression

GEOMETRY 資料類型的欄，或是評估為 GEOMETRY 類型的運算式。

[WITHIN GROUP (ORDER BY sort_expression1 [ASC | DESC] [, sort_expression2 [ASC | DESC] ...])]

選用子句，指定彙總值的排序順序。ORDER BY 子句包含排序運算式的清單。排序運算式是類似於查詢選取清單中的有效排序運算式 (例如欄名稱) 的運算式。您可以指定遞增 (ASC) 或遞減 (DESC) 順序。預設值為 ASC。

傳回類型

MULTIPOINT、MULTILINESTRING、MULTIPOLYGON 或 GEOMETRYCOLLECTION 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom1 或 geom2 為 null，則傳回 null。

如果 aggregate_expression 的所有列都為 null，則傳回 null。

如果 geom1 為 null，則傳回 geom2 的副本。同樣，如果 geom2 為 null，則傳回 geom1 的副本。

如果 geom1 和 geom2 具有不同的 SRID 值，則會傳回錯誤。

如果 aggregate_expression 中的兩個幾何具有不同的 SRID 值，則會傳回錯誤。

如果傳回的幾何大於 GEOMETRY 的大小上限，則會傳回錯誤。

如果 geom1 和 geom2 的維度不同，則傳回錯誤。

如果 aggregate_expression 中的兩個幾何具有不同的維度，則會傳回錯誤。

範例

下列 SQL 會傳回包含兩個輸入幾何的幾何集合。

```
SELECT ST_AsText(ST_Collect(ST_GeomFromText('LINESTRING(0 0,1 1)'),
 ST_GeomFromText('POLYGON((10 10,20 10,10 20,10 10))')));
```

```
st_astext
-----------
 GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),POLYGON((10 10,20 10,10 20,10 10)))
```

下列 SQL 會將表格中的所有幾何圖形收集到幾何集合中。

```
WITH tbl(g) AS (SELECT ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT NULL::geometry UNION ALL
SELECT ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326))
SELECT ST_AsEWKT(ST_Collect(g)) FROM tbl;
```

```
st_astext
-----------
 SRID=4326;GEOMETRYCOLLECTION(POINT(1 2),LINESTRING(0 0,10 0),MULTIPOINT((13 4),(8 5),
(4 4)),POLYGON((0 0,10 0,0 10,0 0)))
```

下列 SQL 會收集資料表中依 id 欄分組並依此 ID 排序的所有幾何。在此範例中，產生的幾何會依 ID 分組，如下所示：

- id 1 – 多點中的點。
- id 2 – multilinestring 中的 linestring。

- id 3 – 幾何集合中的混合子類型。
- id 4 – 多重多邊形中的多邊形。
- id 5 – null，結果為 null。

```
WITH tbl(id, g) AS (SELECT 1, ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT 1, ST_GeomFromText('POINT(4 5)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(10 0,20 -5)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5))', 4326) UNION
 ALL
SELECT 4, ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326) UNION ALL
SELECT 4, ST_GeomFromText('POLYGON((20 20,20 30,30 20,20 20))', 4326) UNION ALL
SELECT 1, NULL::geometry UNION ALL SELECT 2, NULL::geometry UNION ALL
SELECT 5, NULL::geometry UNION ALL SELECT 5, NULL::geometry)
SELECT id, ST_AsEWKT(ST_Collect(g)) FROM tbl GROUP BY id ORDER BY id;
```

```
 id |                                                st_asewkt

----
+-----------------------------------------------------------------------------------------------
  1 | SRID=4326;MULTIPOINT((1 2),(4 5))
  2 | SRID=4326;MULTILINESTRING((0 0,10 0),(10 0,20 -5))
  3 | SRID=4326;GEOMETRYCOLLECTION(MULTIPOINT((13 4),(8 5),(4 4)),MULTILINESTRING((-1
 -1,-2 -2),(-3 -3,-5 -5)))
  4 | SRID=4326;MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((20 20,20 30,30 20,20 20)))
  5 |
```

下列 SQL 會從幾何集合中的資料表收集所有幾何。結果會依 id 的遞減順序排序，然後根據它們的最小和最大 x 座標按字典順序排列。

```
WITH tbl(id, g) AS (
SELECT 1, ST_GeomFromText('POINT(4 5)', 4326) UNION ALL
SELECT 1, ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(10 0,20 -5)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5))', 4326) UNION
 ALL
```

```
SELECT 4, ST_GeomFromText('POLYGON((20 20,20 30,30 20,20 20))', 4326) UNION ALL
SELECT 4, ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326) UNION ALL
SELECT 1, NULL::geometry UNION ALL SELECT 2, NULL::geometry UNION ALL
SELECT 5, NULL::geometry UNION ALL SELECT 5, NULL::geometry)
SELECT ST_AsEWKT(ST_Collect(g) WITHIN GROUP (ORDER BY id DESC, ST_XMin(g), ST_XMax(g)))
 FROM tbl;
```

```
                          st_asewkt

-----------------------------------------------------------------------------------------------
 SRID=4326;GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0 0)),POLYGON((20 20,20 30,30
 20,20 20)),MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5)),MULTIPOINT((13 4),(8 5),(4
 4)),LINESTRING(0 0,10 0),LINESTRING(10 0,20 -5),POINT(1 2),POINT(4 5)
```

ST_Contains

ST_Contains 會在第一個輸入幾何的 2D 投影包含第二個輸入幾何的 2D 投影時傳回 true。如果 B 中的每個點都是 A 中的點，且其內部包含非空白的交集，則幾何 A 包含幾何 B。

ST_Contains(A, B) 與 ST_Within(B, A) 相等。

語法

```
ST_Contains(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom1 比較，以判斷其是否包含在 geom1 中。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否包含第二個多邊形。

```
SELECT ST_Contains(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_contains
-----------
 false
```

ST_ ContainsProperly

如ContainsProperly 果兩個輸入幾何圖形都非空白，且第二個幾何圖形的 2D 投影的所有點都是第一個幾何圖形的 2D 投影的內部點，ST_ 會傳回 true。

語法

```
ST_ContainsProperly(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型不能是 GEOMETRYCOLLECTION。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型不能是 GEOMETRYCOLLECTION。此值會與 geom1 進行比較，以判斷其所有點是否都是 geom1 的內點。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 返回 st_Contains 和 ST_ 的值，ContainsProperly 其中輸入線串與輸入多邊形的內部和邊界相交（但不是其外部）。多邊形包含 linestring，但未正確包含 linestring。

```
WITH tmp(g1, g2)
AS (SELECT ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0))'),
 ST_GeomFromText('LINESTRING(5 5,10 5,10 6,5 5)')) SELECT ST_Contains(g1, g2),
 ST_ContainsProperly(g1, g2)
FROM tmp;
```

```
 st_contains | st_containsproperly
-------------+---------------------
 t           | f
```

ST_ ConvexHull

ST_ ConvexHull 傳回幾何，代表輸入幾何中包含的非空點的凸包。

對於空輸入，產生的幾何與輸入幾何相同。對於所有非空輸入，此函數對輸入幾何的 2D 投影進行操作。但是，輸出幾何的維度取決於輸入幾何的維度。更具體地說，當輸入幾何是非空的 3DM 或 3D 幾何時，會捨棄 m 座標。也就是說，傳回幾何的維度分別為 2D 或 3DZ。如果輸入是非空的 2D 或 3DZ 幾何，則產生的幾何具有相同的維度。

語法

```
ST_ConvexHull(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

凸包上的點數	幾何子類型
0	傳回 geom 的副本。
1	傳回 POINT 子類型。
2	傳回 LINESTRING 子類型。傳回 linestring 的兩個點按字典順序排序。
3 或以上	傳回沒有內環的 POLYGON 子類型。多邊形是順時針方向的，外環的第一個點是環的字典順序最小的點。

範例

以下 SQL 會傳回 linestring 的擴充已知文字 (EWKT) 表示法。在此情況下，傳回的凸包為多邊形。

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('LINESTRING(0 0,1 0,0 1,1 1,0.5 0.5)')))
 as output;
```

```
output
-------------
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

以下 SQL 會傳回 linestring 的 EWKT 表示法。在此情況下，傳回的凸包為 linestring。

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('LINESTRING(0 0,1 1,0.2 0.2,0.6 0.6,0.5
 0.5)'))) as output;
```

```
output
-------------
LINESTRING(0 0,1 1)
```

以下 SQL 會傳回多點的 EWKT 表示法。在此情況下，傳回的凸包為點。

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('MULTIPOINT(0 0,0 0,0 0)'))) as output;
```

```
output
-------------
POINT(0 0)
```

ST_ CoveredBy

如CoveredBy 果第一個輸入幾何圖形的 2D 投影被第二個輸入幾何圖形的 2D 投影覆蓋，ST_ 會傳回 true。如果兩個幾何皆非空白，且 A 中的每個點都是 B 中的點，則幾何 B 涵蓋幾何 A。

ST_ CoveredBy (A,B) 相當於 ST_ 封面 (B,)。A

語法

```
ST_CoveredBy(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom2 比較，以判斷其是否涵蓋於 geom2。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第二個多邊形是否涵蓋第一個多邊形。

```
SELECT ST_CoveredBy(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_coveredby
------------
 true
```

ST_Covers

ST_Covers 會在第一個輸入幾何的 2D 投影涵蓋第二個輸入幾何的 2D 投影時傳回 true。如果兩個幾何皆非空白，且 B 中的每個點都是 A 中的點，則幾何 A 涵蓋幾何 B。

ST_ 封面 (A,B) 等同於 ST_ CoveredBy (B,)。A

語法

```
ST_Covers(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom1 比較，以判斷其是否涵蓋 geom1。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否涵蓋第二個多邊形。

```
SELECT ST_Covers(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_covers
-----------
 false
```

ST_Crosses

如果兩個輸入幾何的 2D 投影互相交叉，ST_Crosses 會傳回 true。

語法

```
ST_Crosses(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否與第二個多點交錯。在此範例中，多點與多邊形的內部和外部相交，這就是 ST_Crosses 傳回 true 的原因。

```
SELECT ST_Crosses (ST_GeomFromText('polygon((0 0,10 0,10 10,0 10,0 0))'),
 ST_GeomFromText('multipoint(5 5,0 0,-1 -1)'));
```

```
st_crosses
-------------
 true
```

以下 SQL 會檢查第一個多邊形是否與第二個多點交錯。在此範例中，多點與多邊形的外部相交，但不與多邊形的內部相交，這就是 ST_Crosses 傳回 false 的原因。

```
SELECT ST_Crosses (ST_GeomFromText('polygon((0 0,10 0,10 10,0 10,0 0))'),
 ST_GeomFromText('multipoint(0 0,-1 -1)'));
```

```
st_crosses
-------------
 false
```

ST_Dimension

ST_Dimension 會傳回輸入幾何的固有維度。「固有維度」是定義在幾何中子類型的維度值。

對於 3DM、3DZ 和 4D 幾何輸入，ST_Dimension 會傳回與 2D 幾何輸入相同的結果。

語法

```
ST_Dimension(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 geom 的固有維度。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回的值	幾何子類型
0	在 geom 是 POINT 或 MULTIPOINT 子類型時傳回
1	在 geom 是 LINESTRING 或 MULTILINESTRING 子類型時傳回。
2	在 geom 是 POLYGON 或 MULTIPOLYGON 子類型時傳回
0	在 geom 是空白的 GEOMETRYCOLLECTION 子類型時傳回
集合元件的最大維度	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會將四個點 LINESTRING 的已知文字 (WKT) 表示法轉換成 GEOMETRY 物件，並傳回 linestring 的維度。

```
SELECT ST_Dimension(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
```

```
st_dimension
-------------
1
```

ST_Disjoint

ST_Disjoint 會在兩個輸入幾何的 2D 投影沒有任何相同點時傳回 True。

語法

```
ST_Disjoint(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否與第二個多邊形斷續。

```
SELECT ST_Disjoint(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(2 2,2 5,5 5,5
 2,2 2))'), ST_Point(4, 4));
```

```
st_disjoint
-----------
 true
```

ST_Distance

對於輸入幾何，ST_Distance 會傳回兩個輸入幾何值的 2D 投影之間的最小歐幾里得距離。

對於 3DM、3DZ、4D 幾何，ST_Distance 會傳回兩個輸入幾何值的 2D 投影之間的歐幾里德距離。

對於輸入地理，ST_Distance 會傳回兩個 2D 點的測地距離。距離的單位是公尺。對於點和空點以外的地理，會傳回錯誤。

語法

```
ST_Distance(geo1, geo2)
```

引數

geo1

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。geo1 的資料類型必須與 geo2 相同。

geo2

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。geo2 的資料類型必須與 geo1 相同。

傳回類型

與輸入幾何或地理相同單位的 DOUBLE PRECISION。

如果 geo1 或 geo2 為 null 或空白，則傳回 null。

如果 geo1 和 geo2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geo1 或 geo2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會傳回兩個多邊形之間的距離。

```
SELECT ST_Distance(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 -3,-2 -1,0 -3,-1 -3))'));
```

```
  st_distance
```

```
-----------
1.4142135623731
```

以下 SQL 使用 GEOGRAPHY 資料類型，傳回柏林的布蘭登堡門和德國國會大廈之間的距離 (以公尺為單位)。

```
SELECT ST_Distance(ST_GeogFromText('POINT(13.37761826722198 52.516411678282445)'),
 ST_GeogFromText('POINT(13.377950831464005 52.51705102546893)'));
```

```
   st_distance
------------------
 74.64129172609631
```

ST_ DistanceSphere

ST_ 會DistanceSphere 傳回位於球面上的兩點幾何圖形之間的距離。

語法

```
ST_DistanceSphere(geom1, geom2)
```

```
ST_DistanceSphere(geom1, geom2, radius)
```

引數

geom1

球面上 GEOMETRY 資料類型的點值，單位為度。點的第一個座標是經度值。點的第二個座標是緯度值。對於 3DZ、3DM 或 4D 幾何，僅使用前兩個座標。

geom2

球面上 GEOMETRY 資料類型的點值，單位為度。點的第一個座標是經度值。點的第二個座標是緯度值。對於 3DZ、3DM 或 4D 幾何，僅使用前兩個座標。

radius

DOUBLE PRECISION 資料類型的球面半徑。如果沒有提供 radius，則球面會預設為地球，且會從橢球的全球測量系統 (WGS) 84 表示法運算半徑。

傳回類型

與半徑單位相同的 DOUBLE PRECISION。如果未提供半徑，則距離以公尺為單位。

如果 geom1 或 geom2 為 Null 或空白，則會傳回 Null。

如果沒有提供 radius，則結果的單位會是沿著地球表面的公尺數。

如果 radius 為負數，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 不是點，則會傳回錯誤。

範例

以下範例 SQL 會計算地球上兩點之間的距離 (以公里為單位)。

```
SELECT ROUND(ST_DistanceSphere(ST_Point(-122, 47), ST_Point(-122.1, 47.1))/ 1000, 0);
```

```
  round
-----------
 13
```

以下範例 SQL 會運算德國三個機場位置間的距離 (公里)：柏林泰格爾機場 (TXL)、慕尼黑國際機場 (MUC) 及法蘭克福國際機場 (FRA)。

```
WITH airports_raw(code,lon,lat) AS (
(SELECT 'MUC', 11.786111, 48.353889) UNION
(SELECT 'FRA', 8.570556, 50.033333) UNION
(SELECT 'TXL', 13.287778, 52.559722)),
airports1(code,location) AS (SELECT code, ST_Point(lon, lat) FROM airports_raw),
airports2(code,location) AS (SELECT * from airports1)
SELECT (airports1.code || ' <-> ' || airports2.code) AS airports,
round(ST_DistanceSphere(airports1.location, airports2.location) / 1000, 0) AS
 distance_in_km
FROM airports1, airports2 WHERE airports1.code < airports2.code ORDER BY 1;
```

```
  airports   | distance_in_km
```

```
-------------+----------------
 FRA <-> MUC |            299
 FRA <-> TXL |            432
 MUC <-> TXL |            480
```

ST_DWithin

ST_DWithin 會在兩個輸入幾何值的 2D 投影之間的歐幾里得距離不大於閾值時傳回 True。

語法

```
ST_DWithin(geom1, geom2, threshold)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

threshold

DOUBLE PRECISION 資料類型的值。這個值的單位是輸入引數。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 threshold 為負值，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查兩個多邊形間的距離是否在五個單位內。

```
SELECT ST_DWithin(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'),5);
```

```
st_dwithin
-----------
 true
```

ST_ EndPoint

ST_ EndPoint 返回輸入線串的最後一個點。結果的空間參考系統識別碼 (SRID) 值與輸入幾何的 SRID 值相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_EndPoint(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空白，則會傳回 Null。

如果 geom 不是 LINESTRING，則會傳回 Null。

範例

下列 SQL 將四點 LINESTRING 的擴充已知文字 (EWKT) 表示法傳回至 GEOMETRY 物件，並傳回 linestring 的終點。

```
SELECT ST_AsEWKT(ST_EndPoint(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0
 5)',4326)));
```

```
st_asewkt
-------------
 SRID=4326;POINT(0 5)
```

ST_Envelope

ST_Enelope 會傳回輸入幾何的最小週框方塊，如下所示。

- 如果輸入幾何為空白，則傳回的幾何為輸入幾何的副本。
- 如果輸入幾何圖形的最小週框方塊退化為一點，則傳回的幾何為一點。
- 如果輸入幾何的最小週框方塊為一維，則會傳回兩點 linestring。
- 如果上述情況都不成立，則函數會傳回順時針方向的多邊形，其頂點是最小週框方塊的邊角。

所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

對於所有非空輸入，此函數對輸入幾何的 2D 投影進行操作。

語法

```
ST_Envelope(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 將四點 LINESTRING 的已知文字 (WKT) 表示法轉換為 GEOMETRY 物件，並傳回一個多邊形，其邊角是最小週框方塊。

```
SELECT ST_AsText(ST_Envelope(ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0
 10,0 0)),LINESTRING(20 10,20 0,10 0))')));
```

```
    st_astext
-----------------------------------
  POLYGON((0 0,0 10,20 10,20 0,0 0))
```

ST_Equals

ST_Equals 會在輸入幾何的 2D 投影在幾何上相等時傳回 True。如果幾何具有相同的點集合，且其內部包含非空白的交集，便可將幾何視為幾何相等。

語法

```
ST_Equals(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom1 比較，以判斷其是否與 geom1 相等。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查兩個多邊形是否幾何相等。

```
SELECT ST_Equals(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_equals
-----------
 false
```

以下 SQL 會檢查兩個 linestring 是否幾何相等。

```
SELECT ST_Equals(ST_GeomFromText('LINESTRING(1 0,10 0)'), ST_GeomFromText('LINESTRING(1
 0,5 0,10 0)'));
```

```
st_equals
-----------
 true
```

ST_ ExteriorRing

ST_ 會ExteriorRing 傳回代表輸入多邊形外環的封閉線串。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_ExteriorRing(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

LINESTRING 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

如果 geom 不是多邊形，則會傳回 Null。

如果 geom 為空，則傳回空的多邊形。

範例

下列 SQL 會將多邊形的外環傳回為封閉的 linestring。

```
SELECT ST_AsEWKT(ST_ExteriorRing(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17
 7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12
 14,15 14,13 11,12 14))')));
```

```
st_asewkt
-----------
 LINESTRING(7 9,8 7,11 6,15 8,16 6,17 7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9)
```

ST_Force2D

ST_Force2D 會傳回輸入幾何的 2D 幾何。對於 2D 幾何，會傳回輸入的副本。對於 3DZ、3DM 和 4D 幾何，ST_Force2D 會將幾何投影至 xy 笛卡爾平面。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force2D(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DZ 幾何傳回 2D 幾何。

```
SELECT ST_AsEWKT(ST_Force2D(ST_GeomFromText('MULTIPOINT Z(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
  MULTIPOINT((0 1),EMPTY,(2 3),(5 6))
```

ST_Force3D

ST_Force3D 是 ST_Force3DZ 的別名。如需詳細資訊，請參閱 ST_Force3DZ。

ST_Force3DM

ST_Force3DM 會傳回輸入幾何的 3DM 幾何。對於 2D 幾何，輸出幾何中非空點的 m 座標均設定為 0。對於 3DM 幾何，會傳回輸入幾何的副本。對於 3DZ 幾何，幾何會投影到 xy 笛卡爾平面，並且輸出幾何中非空點的 m 座標全部設定為 0。對於 4D 幾何，會將幾何投影至 xym 笛卡爾空間。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force3DM(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DZ 幾何傳回 3DM 幾何。

```
SELECT ST_AsEWKT(ST_Force3DM(ST_GeomFromText('MULTIPOINT Z(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
  MULTIPOINT M ((0 1 0),EMPTY,(2 3 0),(5 6 0))
```

ST_Force3DZ

ST_Force3DZ 會從輸入幾何傳回 3DZ 幾何。對於 2D 幾何，輸出幾何中非空點的 z 座標均設定為 0。對於 3DM 幾何，幾何會投影到 xy 笛卡爾平面，並且輸出幾何中非空點的 z 座標全部設定為 0。對於 3DZ 幾何，會傳回輸入幾何的副本。對於 4D 幾何，會將幾何投影至 xyz 笛卡爾空間。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force3DZ(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DM 幾何傳回 3DZ 幾何。

```
SELECT ST_AsEWKT(ST_Force3DZ(ST_GeomFromText('MULTIPOINT M(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
  MULTIPOINT Z ((0 1 0),EMPTY,(2 3 0),(5 6 0))
```

ST_Force4D

ST_Force4D 會傳回輸入幾何的 4D 幾何。對於 2D 幾何，輸出幾何中非空點的 z 和 m 座標均設定為 0。對於 3DM 幾何，輸出幾何中非空點的 z 座標均設定為 0。對於 3DZ 幾何，輸出幾何中非空點的 m 座標均設定為 0。對於 4D 幾何，會傳回輸入幾何的副本。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force4D(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DM 幾何傳回 4D 幾何。

```
SELECT ST_AsEWKT(ST_Force4D(ST_GeomFromText('MULTIPOINT M(0 1 2, EMPTY, 2 3 4, 5 6
 7)')));
```

```
st_asewkt
-----------
  MULTIPOINT ZM ((0 1 0 2),EMPTY,(2 3 0 4),(5 6 0 7))
```

ST_ GeoHash

ST_ GeoHash 返回具有指定精確度的輸入點的geohash表示。預設精確度為 20。如需 geohash 定義的相關資訊，請參閱 Wikipedia 中的 Geohash。

語法

```
ST_GeoHash(geom)
```

```
ST_GeoHash(geom, precision)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

precision

INTEGER 資料類型的值。預設值為 20。

傳回類型

GEOMETRY

函數會傳回輸入點的 geohash 表示。

如果輸入點為空，該函數傳回 null。

如果輸入幾何不是點，則函數會傳回錯誤。

範例

以下 SQL 會傳回輸入點的 geohash 表示法。

```
SELECT ST_GeoHash(ST_GeomFromText('POINT(45 -45)'), 25) AS geohash;
```

```
          geohash
--------------------------
 m0000000000000000000gzz
```

下列 SQL 傳回 null，因為輸入點是空的。

```
SELECT ST_GeoHash(ST_GeomFromText('POINT EMPTY'), 10) IS NULL AS result;
```

```
 result
---------
 true
```

ST_ GeogFromText

ST_ GeogFromText 構造從一個知名的文本（WKT）或輸入地理的擴展知名文本（EWKT）表示的地理對象。

語法

```
ST_GeogFromText(wkt_string)
```

引數

wkt_string

VARCHAR 資料類型的值，其為幾何的 WKT 或 EWKT 表示法。

傳回類型

GEOGRAPHY

如果 SRID 值設定為輸入中提供的值。如果未提供 SRID，則會設定為 4326。

如果 wkt_string 為 null，則傳回 null。

如果 wkt_string 無效，則會傳回錯誤。

範例

以下 SQL 會使用 SRID 值從地理物件建構多邊形。

```
SELECT ST_AsEWKT(ST_GeogFromText('SRID=4324;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))'));
```

```
  st_asewkt
-----------------------------------------------
 SRID=4324;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))
```

以下 SQL 從地理物件建構一個多邊形。SRID 的值設定為 4326。

```
SELECT ST_AsEWKT(ST_GeogFromText('POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))'));
```

```
 st_asewkt
-----------------------------------------------
 SRID=4326;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))
```

ST_ 白板 GeogFrom

ST_ GeogFrom WKB 構造從輸入地理的十六進制知名二進制（WKB）表示的地理對象。

語法

```
ST_GeogFromWKB(wkb_string)
```

引數

wkb_string

VARCHAR 資料類型的值，其為地理的十六進位 WKB 表示法。

傳回類型

GEOGRAPHY

如果提供 SRID 值，則會將設為提供的值。如果未提供 SRID，則會設定為 4326。

如果 wkb_string 為 null，則傳回 null。

如果 wkb_string 無效，則會傳回錯誤。

範例

下列 SQL 從十六進位 WKB 值建構地理。

```
SELECT
 ST_AsEWKT(ST_GeogFromWKB('010300000001000000050000000000000000000000000000000000000000000000000000000000000000
```

```
 st_asewkt
------------------------------------------
 SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_GeometryN

ST_GeometryN 會傳回由輸入幾何的輸入索引指向的幾何，如下所示。

- 如果輸入是點、linestring 或多邊形，則若索引等於一 (1)，傳回幾何，若索引不是一 (1)，傳回 Null。
- 如果輸入是 multipoint、multilinestring、multipolygon 或幾何集合，則會傳回由輸入索引指向的點、linestring、多邊形或幾何集合。

索引是以一開始。結果的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_GeometryN(geom, index)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

索引

資料類型 INTEGER 的值，代表以一開始的索引位置。

傳回類型

GEOMETRY

如果 geom 或 index 為 Null，則會傳回 Null。

如果 index 超出範圍，則會傳回錯誤。

範例

下列 SQL 會傳回幾何集合中的幾何。

```
WITH tmp1(idx) AS (SELECT 1 UNION SELECT 2),
tmp2(g) AS (SELECT ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0
 0)),LINESTRING(20 10,20 0,10 0))'))
SELECT idx, ST_AsEWKT(ST_GeometryN(g, idx)) FROM tmp1, tmp2 ORDER BY idx;
```

```
 idx |          st_asewkt
-----+-----------------------------
   1 | POLYGON((0 0,10 0,0 10,0 0))
   2 | LINESTRING(20 10,20 0,10 0)
```

ST_ GeometryType

ST_ 以字串形式GeometryType 傳回輸入幾何的子類型。

對於 3DM、3DZ 和 4D 幾何輸入，ST_ 會GeometryType 傳回與 2D 幾何輸入相同的結果。

語法

```
ST_GeometryType(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

VARCHAR 表示 geom 的子類型。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回的字串值	幾何子類型
ST_Point	在 geom 是 POINT 子類型時傳回
ST_LineString	在 geom 是 LINESTRING 子類型時傳回
ST_Polygon	在 geom 是 POLYGON 子類型時傳回
ST_MultiPoint	在 geom 是 MULTIPOINT 子類型時傳回
ST_MultiLineString	在 geom 是 MULTILINESTRING 子類型時傳回
ST_MultiPolygon	在 geom 是 MULTIPOLYGON 子類型時傳回
ST_GeometryCollection	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會傳回輸入 linestring 幾何的子類型。

```
SELECT ST_GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
```

```
st_geometrytype
-------------
 ST_LineString
```

ST _ EWKB GeomFrom

ST_ GeomFrom EWKB 構造從一個輸入幾何的擴展眾所周知的二進制（EWKB）表示的幾何對象。

ST_ GeomFrom EWKB 接受以 WKB 和 EWKB 十六進位格式撰寫的 3DZ、3 分 M 和 4D 幾何圖形。

語法

```
ST_GeomFromEWKB(ewkb_string)
```

引數

ewkb_string

VARCHAR 資料類型的值，其為幾何的十六進位 EWKB 表示法。

傳回類型

GEOMETRY

如果 ewkb_string 為 Null，則會傳回 Null。

如果 ewkb_string 無效，則會傳回錯誤。

範例

以下 SQL 會從 EWKB 值建構多邊形，並傳回多邊形的 EWKT 表示法。

```
SELECT
 ST_AsEWKT(ST_GeomFromEWKB('0103000020E6100000010000000500000000000000000000000000000000000000000
```

```
 st_asewkt
-------------------------------
 SRID=4326;POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST-尤克特 GeomFrom

ST_ GeomFrom EWKT 會從輸入幾何圖形的延伸已知文字 (EWKT) 表現法建構幾何圖形物件。

ST_ GeomFrom EWKT 接受 3DZ、3DM 和 4D，其中幾何類型的前綴分別為 Z、M 或 ZM。

語法

```
ST_GeomFromEWKT(ewkt_string)
```

引數

ewkt_string

VARCHAR 資料類型的值，或計算結果為 VARCHAR 類型的運算式，也就是幾何的 EWKT 表示法。

您可以使用 WKT 關鍵字 EMPTY 來指定空點、具有空點的多點或具有空點的幾何集合。以下範例會建立空點。

```
ST_GeomFromEWKT('SRID=4326;POINT EMPTY');
```

傳回類型

GEOMETRY

如果 ewkt_string 為 null，則傳回 null。

如果 ewkt_string 無效，則會傳回錯誤。

範例

下列 SQL 會從 EWKT 值建構 multilinestring，並傳回幾何。它也會傳回幾何的 ST_AsEWKT 結果。

```
SELECT ST_GeomFromEWKT('SRID=4326;MULTILINESTRING((1 0,1 0),(2 0,3 0),(4 0,5 0,6 0))')
 as geom, ST_AsEWKT(geom);
```

```
                                                    geom

                                          |                          st_asewkt

-----------------------------------------------------------------------------------------------
+-----------------------------------------------------------
 0105000020E6100000030000000102000000020000000000000000F03F0000000000000000000000000000F03F00
 | SRID=4326;MULTILINESTRING((1 0,1 0),(2 0,3 0),(4 0,5 0,6 0))
```

ST_ GeomFromGeoHash

ST_ 從輸入幾何的 Geohash 表示法GeomFromGeoHash 構造一個幾何對象。ST_ 會 GeomFromGeoHash 傳回空間參考識別碼 (SRID) 為零 (0) 的二維 (2D) 幾何圖形。如需 geohash 格式的相關資訊，請參閱 Wikipedia 中的 Geohash。

語法

```
ST_GeomFromGeoHash(geohash_string)
```

```
ST_GeomFromGeoHash(geohash_string, precision)
```

引數

geohash_string

VARCHAR 資料類型的值，或計算結果為 VARCHAR 類型的運算式，也就是幾何的 geohash 表示法。

precision

INTEGER 資料類型的值，代表 geohash 的精確度。該值是用作精確度的 geohash 的字元數。如果未指定該值，則小於零或大於 geohash_string 長度，則會使用 geohash_string 長度。

傳回類型

GEOMETRY

如果 geohash_string 為 null，則傳回 null。

如果 geohash_string 無效，則會傳回錯誤。

範例

下列 SQL 會傳回高精確度的多邊形。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0'));
```

```
 st_asewkt
-----------------------
 POLYGON((-115.172816 36.114646,-115.172816 36.114646,-115.172816 36.114646,-115.172816
 36.114646,-115.172816 36.114646))
```

以下 SQL 會傳回高精確度的點。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz00'));
```

```
 st_asewkt
-----------------------
 POINT(-115.172816 36.114646)
```

下列 SQL 會傳回低精確度的多邊形。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qq'));
```

```
 st_asewkt
-----------------------
 POLYGON((-115.3125 35.15625,-115.3125 36.5625,-113.90625 36.5625,-113.90625
 35.15625,-115.3125 35.15625))
```

下列 SQL 會傳回精確度為 3 的多邊形。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxncgyy4d0dbxqz0', 3));
```

```
 st_asewkt
-----------------------
 POLYGON((-115.3125 35.15625,-115.3125 36.5625,-113.90625 36.5625,-113.90625
 35.15625,-115.3125 35.15625))
```

JSON GeomFromGeo

ST_ GeomFromGeo JSON 構造從輸入幾何的 GeoJSON 表示形式的幾何對象。如需 GeoJSON 格式的相關資訊，請參閱 Wikipedia 中的 GeoJSON。

如果至少有一個點具有三個或更多座標，則產生的幾何圖形為 3DZ，其中對於僅具有兩個座標的點，Z 分量為零。如果輸入 GeoJSON 中的所有點都包含兩個坐標或為空，則 ST_ GeomFromGeo JSON 返回一個二維幾何圖形。傳回幾何的空間參考系統識別碼 (SRID) 一律為 4326。

語法

```
ST_GeomFromGeoJSON(geojson_string)
```

引數

geojson_string

VARCHAR 資料類型的值，或計算結果為 VARCHAR 類型的運算式，也就是幾何的 GeoJSON 表示法。

傳回類型

GEOMETRY

如果 geojson_string 為 null，則傳回 null。

如果 geojson_string 無效，則會傳回錯誤。

範例

以下 SQL 傳回輸入 GeoJSON 中表示的二維幾何。

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"Point","coordinates":[1,2]}'));
```

```
 st_asewkt
-----------------------
 SRID=4326;POINT(1 2)
```

以下 SQL 傳回輸入 GeoJSON 中表示的 3DZ 幾何。

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"LineString","coordinates":[[1,2,3],
[4,5,6],[7,8,9]]}'));
```

```
 st_asewkt
----------------------------------------
 SRID=4326;LINESTRING Z (1 2 3,4 5 6,7 8 9)
```

當輸入 GeoJSON 中只有一個點具有三個座標而所有其他點都具有兩個座標時，以下 SQL 傳回 3DZ 幾何。

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"Polygon","coordinates":[[[0, 0],[0, 1,
 8],[1, 0],[0, 0]]]}'));
```

```
 st_asewkt
-----------------------------------------------
 SRID=4326;POLYGON Z ((0 0 0,0 1 8,1 0 0,0 0 0))
```

ST_ GeomFromGeoSquare

ST_ 會GeomFromGeoSquare 傳回涵蓋由輸入土平方值所表示區域的幾何圖形。傳回的幾何永遠是二維的。若要計算 geosquare 值，請參閱 ST_ GeoSquare。

語法

```
ST_GeomFromGeoSquare(geosquare)
```

```
ST_GeomFromGeoSquare(geosquare, max_depth)
```

引數

geosquare

BIGINT 資料類型的值或計算結果為 geosquare 值之 BIGINT 類型的運算式，該值描述在初始域上進行的細分序列以達到所需的平方。此值由 ST_ GeoSquare 計算。

max_depth

INTEGER 資料類型的值，代表對初始域進行的域細分的最大數量。值必須等於或大於 1。

傳回類型

GEOMETRY

如果 geosquare 無效，則函數會傳回錯誤。

如果輸入 max_depth 不在範圍內，則函數會傳回錯誤。

範例

下列 SQL 會從 geosquare 值傳回幾何。

```
SELECT ST_AsText(ST_GeomFromGeoSquare(797852));
```

```
 st_astext
-----------------------------------------------------------------------------------------------------
 POLYGON((13.359375 52.3828125,13.359375 52.734375,13.7109375 52.734375,13.7109375
 52.3828125,13.359375 52.3828125))
```

下列 SQL 會從 geosquare 值和最大深度 3 傳回幾何。

```
SELECT ST_AsText(ST_GeomFromGeoSquare(797852, 3));
```

```
 st_astext
-------------------------------------
 POLYGON((0 45,0 90,45 90,45 45,0 45))
```

以下 SQL 首先透過指定 x 座標為經度、y 座標為緯度 (-122.3, 47.6) 來計算西雅圖的 geosquare 值。然後它傳回 geosquare 的多邊形。雖然輸出是二維幾何，但它可以用來計算經度和緯度的空間資料。

```
SELECT ST_AsText(ST_GeomFromGeoSquare(ST_GeoSquare(ST_Point(-122.3, 47.6))));
```

```
 st_astext
----------------------------------------------------------------------------------------------------
POLYGON((-122.335167014971 47.6080129947513,-122.335167014971
 47.6080130785704,-122.335166931152 47.6080130785704,-122.335166931152
 47.6080129947513,-122.335167014971 47.6080129947513))
```

ST_ GeomFromText

ST_ 從輸入幾何圖形的已知文字 (WKT) 表現法GeomFromText 建構幾何圖形物件。

ST_ GeomFromText 接受 3DZ、3DM 和 4D，其中幾何類型分別以 Z、M 或 ZM 作為前綴。

語法

```
ST_GeomFromText(wkt_string)
```

```
ST_GeomFromText(wkt_string, srid)
```

引數

wkt_string

VARCHAR 資料類型的值，其為幾何的 WKT 表示法。

您可以使用 WKT 關鍵字 EMPTY 來指定空點、具有空點的多點或具有空點的幾何集合。下列範例使用一個空點和一個非空點建立多點。

```
ST_GeomFromEWKT('MULTIPOINT(1 0,EMPTY)');
```

srid

INTEGER 資料類型的值，其為空間參考識別碼 (SRID)。如果有提供 SRID 值，則傳回的幾何便會包含此 SRID 值。否則，傳回幾何的 SRID 值會設為零 (0)。

傳回類型

GEOMETRY

如果 wkt_string 或 srid 為 Null，則會傳回 Null。

如果 srid 為負值，則會傳回 Null。

如果 wkt_string 無效，則會傳回錯誤。

如果 srid 無效，則會傳回錯誤。

範例

以下 SQL 會從 WKT 表示法和 SRID 值建構幾何物件。

```
SELECT ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))',4326);
```

```
st_geomfromtext
--------------------------------
0103000020E61000000100000005000000000000000000000000000000000000000000000000000000000000000000F
```

ST_ 白板 GeomFrom

ST_ GeomFrom WKB 構造從一個輸入幾何的十六進制知名二進制（WKB）表示的幾何對象。

ST_ GeomFrom WKB 接受以 WKB 十六進位格式撰寫的 3DZ、3 分 M 和 4D 幾何圖形。

語法

```
ST_GeomFromWKB(wkb_string)
```

```
ST_GeomFromWKB(wkb_string, srid)
```

引數

wkb_string

VARCHAR 資料類型的值，其為幾何的十六進位 WKB 表示法。

srid

INTEGER 資料類型的值，其為空間參考識別碼 (SRID)。如果有提供 SRID 值，則傳回的幾何便會包含此 SRID 值。否則，傳回幾何的 SRID 值會設為 0。

傳回類型

GEOMETRY

如果 wkb_string 或 srid 為 Null，則會傳回 Null。

如果 srid 為負值，則會傳回 Null。

如果 wkb_string 無效，則會傳回錯誤。

如果 srid 無效，則會傳回錯誤。

範例

以下 SQL 會從 WKB 值建構多邊形，並傳回多邊形的 WKT 表示法。

```
SELECT
 ST_AsText(ST_GeomFromWKB('010300000001000000050000000000000000000000000000000000000000000
```

```
 st_astext
-------------------------------
 POLYGON((0 0,0 1,1 1,1 0,0 0))
```

ST_ GeoSquare

ST_ GeoSquare 遞歸地將域（[-180，180]，[-90，90]）分為相等的平方區域，稱為土平方到指定的深度。細分以提供的點的位置為基礎。包含該點的其中一個 geosquare 會在每個步驟細分，直到達到最大深度為止。這個 geosquare 的選擇是穩定的，亦即函數結果僅取決於輸入參數。此函數傳回一個唯一值，用於識別該點所在的最終 geosquare。

ST_ GeoSquare 接受一個點，其中 x 座標代表經度，y 座標代表緯度。經度和緯度分別限制為 [-180, 180] 和 [-90, 90]。ST_ 的輸出GeoSquare 可以用作ST_ GeomFromGeoSquare函數的輸入。

地球赤道圓周有 360° 的弧線，分為兩個半球 (東半球和西半球)，每個半球都有從 0° 子午線開始的 180° 經線 (子午線)。按照慣例，當投影到笛卡爾平面上的 x 軸時，東經度是「+」(正) 座標，當投影到

笛卡爾平面上的 x 軸時，西經度是「-」(負) 座標。地球赤道週長 0° 的南北各有 90° 緯度線，每條緯線都與地球赤道週長 0° 平行。依照慣例，投影到笛卡爾平面時，北緯線與「+」(正) y 軸相交，投影到笛卡爾平面時，南緯線與「-」(負) y 軸相交。將經線和緯線相交形成的球形網格轉換為投影到笛卡爾平面上的網格，該網格在笛卡爾平面上具有標準的正負 x 座標和正負 y 座標。

ST_ 的目的GeoSquare 是使用相同的代碼值標記或標記關閉點。位於相同 geosquare 的點會接收相同的代碼值。Geosquare 用於將地理座標 (緯度和經度) 編碼為整數。較大的區域被劃分為網格，以在地圖上以不同的解析度描繪區域。Geosquare 可用於空間索引、空間分級、鄰近搜尋、位置搜尋以及建立唯一的地點識別碼。ST_ GeoHash 函數遵循類似的過程，將區域劃分為網格，但具有不同的編碼方式。

語法

```
ST_GeoSquare(geom)
```

```
ST_GeoSquare(geom, max_depth)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 POINT 子類型的運算式。點的 x 座標 (經度) 必須在以下範圍內：-180 - 180。點的 y 座標 (緯度) 必須在以下範圍內：-90 - 90。

max_depth

INTEGER 資料類型的值。包含該點的域被遞歸細分的最大次數。值必須為介於 1 到 32 之間的整數。預設值為 32。子實際最終的細分數小於或等於指定的 max_depth。

傳回類型

BIGINT

此函數傳回一個唯一值，用於識別輸入點所在的最終 geosquare。

如果輸入 geom 不是點，則函數會傳回錯誤。

如果輸入點為空，則傳回值對 ST_ GeomFromGeoSquare 函數不是有效輸入。使用該ST_ IsEmpty函數可防止以空點對 ST_ GeoSquare 的調用。

如果輸入點不在範圍內，則函數會傳回錯誤。

如果輸入 max_depth 超出範圍，則函數會傳回錯誤。

範例

以下 SQL 會從輸入點傳回 geosquare。

```
SELECT ST_GeoSquare(ST_Point(13.5, 52.5));
```

```
  st_geosquare
-----------------------
 -4410772491521635895
```

下列 SQL 從最大深度為 10 的輸入點傳回 geosquare。

```
SELECT ST_GeoSquare(ST_Point(13.5, 52.5), 10);
```

```
 st_geosquare
--------------
 797852
```

ST_ N InteriorRing

ST_ InteriorRing N 會傳回對應於索引位置處輸入多邊形內環的封閉線串。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_InteriorRingN(geom, index)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

索引

INTEGER 資料類型的值，代表以一開始索引的環形位置。

傳回類型

LINESTRING 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 或 index 為 Null，則會傳回 Null。

如果 index 超出範圍，則會傳回 Null。

如果 geom 不是多邊形，則會傳回 Null。

如果 geom 為空多邊形，則傳回 null。

範例

下列 SQL 會將多邊形的第二個環傳回為封閉的 linestring。

```
SELECT ST_AsEWKT(ST_InteriorRingN(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17
 7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12
 14,15 14,13 11,12 14))'),2));
```

```
st_asewkt
-----------
 LINESTRING(12 14,15 14,13 11,12 14)
```

ST_Intersects

ST_Intersects 會在兩個輸入幾何的 2D 投影至少擁有一個相同點時傳回 True。

語法

```
ST_Intersects(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否和第二個多邊形交集。

```
SELECT ST_Intersects(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(2 2,2 5,5 5,5
 2,2 2))'), ST_GeomFromText('MULTIPOINT((4 4),(6 6))'));
```

```
st_intersects
-------------
 true
```

ST_Intersection

ST_Intersection 會傳回代表兩個幾何的點集交集的幾何。也就是說，它會傳回兩個輸入幾何之間共用的部分。

語法

```
ST_Intersection(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom1 和 geom2 不共用任何空間 (它們不相交)，則傳回空的幾何。

如果 geom1 或 geom2 為空，則會傳回空的幾何。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

如果 geom1 或 geom2 不是二維 (2D) 幾何，則會傳回錯誤。

範例

下列 SQL 會傳回表示兩個輸入幾何交集的非空幾何。

```
SELECT ST_AsEWKT(ST_Intersection(ST_GeomFromText('polygon((0 0,100 100,0 200,0 0))'),
 ST_GeomFromText('polygon((0 0,10 0,0 10,0 0))')));
```

```
        st_asewkt
-------------------------
 POLYGON((0 0,0 10,5 5,0 0))
```

當傳遞不連接 (非相交) 的輸入幾何時，下列 SQL 會傳回空的幾何。

```
SELECT ST_AsEWKT(ST_Intersection(ST_GeomFromText('linestring(0 100,0 0)'),
 ST_GeomFromText('polygon((1 0,10 0,1 10,1 0))')));
```

```
    st_asewkt
------------------
 LINESTRING EMPTY
```

ST_ 逆時鐘 IsPolygon

ST_ IsPolygon CCW 如果輸入多邊形或多重多邊形的 2D 投影是逆時針方向返回 true。如果輸入幾何為點、linestring、多點或 multilinestring，則傳回 true。對於幾何圖形集合，如果集合中的所有幾何圖形都是逆時鐘方向，ST_ IsPolygon CCW 會傳回 true。

語法

```
ST_IsPolygonCCW(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否逆時針方向。

```
SELECT ST_IsPolygonCCW(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17 7,17 10,18
 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12 14,15 14,13
 11,12 14))'));
```

```
 st_ispolygonccw
----------
 true
```

ST_ 密西西比 IsPolygon

如果輸入多邊形或多重多邊形的 2D 投影是順時針方向，則 ST_ IsPolygon CW 返回 true。如果輸入幾何為點、linestring、多點或 multilinestring，則傳回 true。對於幾何圖形集合，如果集合中的所有幾何圖形均為順時針方向，則 ST_ IsPolygon CW 會傳回 true。

語法

```
ST_IsPolygonCW(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否順時針方向。

```
SELECT ST_IsPolygonCW(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17 7,17 10,18
 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12 14,15 14,13
 11,12 14))'));
```

```
 st_ispolygonccw
----------
 true
```

ST_ IsClosed

如果關閉輸入幾何圖形的 2D 投影，ST_ 會IsClosed 傳回 true。下列規則定義了封閉幾何：

- 輸入幾何是點或 multipoint。
- 輸入幾何是 linestring，且 linestring 的起始點和結束點一致。
- 輸入幾何是非空白的 multilinestring，且其所有的 linestring 都是封閉的。
- 輸入幾何是非空白的多邊形，所有多邊形的環都是非空白的，且其所有環的起始點和結束點都一致。
- 輸入幾何是非空白的 multipolygon，且其所有的多邊形都是封閉的。

- 輸入幾何是非空白的幾何集合，且其所有的元件都是封閉的。

語法

```
ST_IsClosed(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 是空點，則傳回 false。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否是封閉的。

```
SELECT ST_IsClosed(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_isclosed
-----------
 true
```

ST_ IsCollection

如果輸入幾何是下列其中一種子類型，ST_ 會IsCollection 傳回
true：GEOMETRYCOLLECTION、MULTIPOINTMULTILINESTRING、或。MULTIPOLYGON

語法

```
ST_IsCollection(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否是集合。

```
SELECT ST_IsCollection(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_iscollection
-----------
 false
```

ST_ IsEmpty

ST_ 如果輸入幾何圖形為空，則IsEmpty 返回 true。如果幾何至少包含一個非空點，則該幾何不是空的。

ST_ 如果輸入幾何至少有一個非空點，則IsEmpty 返回 true。

語法

```
ST_IsEmpty(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查指定多邊形是否空白。

```
SELECT ST_IsEmpty(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_isempty
-----------
 false
```

ST_ IsRing

ST_ 如果輸入線串是一個環，則IsRing 返回 true。如果 linestring 是封閉且簡單的，則為環。

語法

```
ST_IsRing(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。幾何必須是 LINESTRING。

傳回類型

BOOLEAN

如果 geom 不是 LINESTRING，則會傳回錯誤。

範例

下列 SQL 會檢查指定的 linestring 是否為環。

```
SELECT ST_IsRing(ST_GeomFromText('linestring(0 0, 1 1, 1 2, 0 0)'));
```

```
st_isring
-----------
 true
```

ST_ IsSimple

如果輸入幾何圖形的 2D 投影很簡單，則 ST_ 會IsSimple 傳回 true。如需簡單幾何定義的相關資訊，請參閱幾何簡單性。

語法

```
ST_IsSimple(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會檢查指定的 linestring 是否簡單。在這個範例中，它並不簡單，因為它具有自我相交。

```
SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(0 0,10 0,5 5,5 -5)'));
```

```
 st_issimple
-----------
 false
```

ST_ IsValid

如果輸入幾何圖形的 2D 投影有效，ST_ 會IsValid 傳回 true。如需有效幾何定義的相關資訊，請參閱幾何有效性。

語法

```
ST_IsValid(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查指定多邊形是否有效。在此範例中，多邊形無效，因為多邊形的內部並未簡單地連接。

```
SELECT ST_IsValid(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(5 0,10 5,5 10,0
 5,5 0))'));
```

```
 st_isvalid
------------
 false
```

ST_Length

對於線性幾何，ST_Length 傳回 2D 投影的笛卡爾長度。長度單位與表示輸入幾何座標的單位相同。對於點、multipoint 和面積幾何，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中幾何的長度總和。

對於地理，ST_Length 會傳回在球體 (由 SRID 決定) 上所計算輸入線性地理之 2D 投影的測地線長度。長度的單位是公尺。對於點、multipoint 和面積地理，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中地理的長度總和。

語法

```
ST_Length(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

DOUBLE PRECISION

如果 geo 為 null，則傳回 null。

如果找不到 SRID 值，則會傳回錯誤。

範例

下列 SQL 會傳回 multilinestring 的笛卡爾長度。

```
SELECT ST_Length(ST_GeomFromText('MULTILINESTRING((0 0,10 0,0 10),(10 0,20 0,20
 10))'));
```

```
st_length
--------------------------------
  44.142135623731
```

以下 SQL 會傳回幾何中 linestring 的長度。

```
SELECT ST_Length(ST_GeogFromText('SRID=4326;LINESTRING(5 0,6 0,4 0)'));
```

```
 st_length
```

```
------------------
 333958.472379804
```

下列 SQL 會傳回地理中某個點的長度。

```
SELECT ST_Length(ST_GeogFromText('SRID=4326;POINT(4 5)'));
```

```
 st_length
-----------
 0
```

ST_ LengthSphere

ST_ 會LengthSphere 傳回以公尺為單位的線性幾何圖形的長度。對於點、多點和面積幾何圖形，ST LengthSphere _ 會傳回 0。對於幾何圖形集合，ST_ 會LengthSphere 傳回集合中線性幾何圖形的總長度 (以公尺為單位)。

ST_ LengthSphere 會將輸入幾何圖形的每個點的座標解譯為經度和緯度 (以度為單位)。對於 3DZ、3DM 或 4D 幾何，僅使用前兩個座標。

語法

```
ST_LengthSphere(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

DOUBLE PRECISION 長度的單位是公尺。長度計算是根據地球的球形模型，其半徑是世界大地測量系統 (WGS) 84 地球橢球模型的地球平均半徑。

如果 geom 為 Null，則會傳回 Null。

範例

下列範例 SQL 會計算 linestring 長度 (以公尺為單位)。

```
SELECT ST_LengthSphere(ST_GeomFromText('LINESTRING(10 10,45 45)'));
```

```
 st_lengthsphere
------------------
 5127736.08292556
```

ST_Length2D

ST_Length2D 是 ST_Length 的別名。如需詳細資訊，請參閱 ST_Length。

ST_ LineFromMultiPoint

ST_ LineFromMultiPoint 會從輸入多點幾何圖形傳回線串。點的順序會保留下來。所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_LineFromMultiPoint(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 MULTIPOINT。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則傳回 LINESTRING。

如果 geom 包含空點，則會忽略這些空點。

如果 geom 不是 MULTIPOINT，則會傳回錯誤。

範例

下列 SQL 會從 multipoint 建立 linestring。

```
SELECT ST_AsEWKT(ST_LineFromMultiPoint(ST_GeomFromText('MULTIPOINT(0 0,10 0,10 10,5 5,0
 5)',4326)));
```

```
 st_asewkt
--------------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 5,0 5)
```

ST_ LineInterpolatePoint

ST_ 沿著直線LineInterpolatePoint 返回一個點，距離直線的起點有分數的距離。

若要確定點相等，ST_ 會對輸入幾何圖形的 2D 投影進LineInterpolatePoint 行操作。如果輸入幾何為空，則會以與輸入相同的維度傳回其副本。對於 3DZ、3DM 和 4D 幾何，z 或 m 座標是點所在區段的 z 或 m 座標的平均值。

語法

```
ST_LineInterpolatePoint(geom, fraction)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型是 LINESTRING。

fraction

DOUBLE PRECISION 資料類型的值，表示沿線的 linestring 的點位置。該值是介於 0-1 (含) 範圍內的分數。

傳回類型

POINT 子類型的 GEOMETRY。

如果 geom 或 fraction 為 null，則傳回 null。

如果 geom 為空，則傳回空點。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 fraction 超出範圍，則會傳回錯誤。

如果 geom 不是 linestring，則會傳回錯誤。

範例

下列 SQL 會傳回沿 linestring 中間的點。

```
SELECT ST_AsEWKT(ST_LineInterpolatePoint(ST_GeomFromText('LINESTRING(0 0, 5 5, 7 7, 10
 10)'), 0.50));
```

```
st_asewkt
-----------
 POINT(5 5)
```

下列 SQL 會傳回沿 linestring 90% 的點。

```
SELECT ST_AsEWKT(ST_LineInterpolatePoint(ST_GeomFromText('LINESTRING(0 0, 5 5, 7 7, 10
 10)'), 0.90));
```

```
st_asewkt
-----------
 POINT(9 9)
```

ST_M

ST_M 會傳回輸入點的 m 座標。

語法

```
ST_M(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

m 座標的 DOUBLE PRECISION 值。

如果 point 是 Null，則會傳回 Null。

如果 point 是 2D 或 3DZ 點，則傳回 null。

如果 point 是空點，則傳回 null。

如果 point 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回 3DZ 幾何中點的 m 座標。

```
SELECT ST_M(ST_GeomFromEWKT('POINT M (1 2 3)'));
```

```
st_m
-----------
 3
```

以下 SQL 會傳回 4D 幾何中點的 m 座標。

```
SELECT ST_M(ST_GeomFromEWKT('POINT ZM (1 2 3 4)'));
```

```
st_m
-----------
 4
```

ST_ MakeEnvelope

ST_ MakeEnvelope 會傳回幾何圖形，如下所示：

- 如果輸入座標指定了一個點，則傳回的幾何為一個點。
- 如果輸入座標指定了直線，則傳回的幾何為 linestring。
- 否則，傳回的幾何為多邊形，其中輸入座標會指定方塊的左下角和右上角。

如果提供，則傳回幾何的空間參考系統識別碼 (SRID) 值將設定為輸入 SRID 值。

語法

```
ST_MakeEnvelope(xmin, ymin, xmax, ymax)
```

```
ST_MakeEnvelope(xmin, ymin, xmax, ymax, srid)
```

引數

xmin

DOUBLE PRECISION 資料類型的值。此值是方塊左下角的第一個座標。

ymin

DOUBLE PRECISION 資料類型的值。此值是方塊左下角的第二個座標。

xmax

DOUBLE PRECISION 資料類型的值。此值為方塊右上角的第一個座標。

ymax

DOUBLE PRECISION 資料類型的值。此值為方塊的右上角的第二個座標。

srid

INTEGER 資料類型的值，表示空間參考系統識別碼 (SRID)。如果未提供 SRID 值，則會將其設定為零。

傳回類型

POINT、LINESTRING 或 POLYGON 子類型的 GEOMETRY。

如果未設定 srid，則傳回幾何的 SRID 會設定為 srid 或零。

如果 xmin、ymin、xmax、ymax 或 srid 為 null，則傳回 null。

如果 srid 為負值，則會傳回錯誤。

範例

下列 SQL 會傳回一個多邊形，表示由四個輸入座標值定義的包絡線。

```
SELECT ST_AsEWKT(ST_MakeEnvelope(2,4,5,7));
```

```
 st_astext
---------------
 POLYGON((2 4,2 7,5 7,5 4,2 4))
```

以下 SQL 會傳回一個多邊形，表示由四個輸入座標值和 SRID 值定義的包絡線。

```
SELECT ST_AsEWKT(ST_MakeEnvelope(2,4,5,7,4326));
```

```
 st_astext
----------------------------------
 SRID=4326;POLYGON((2 4,2 7,5 7,5 4,2 4))
```

ST_ MakeLine

ST_ 從輸入幾何圖MakeLine 形建立線串。

傳回幾何的維度與輸入幾何的維度相同。兩個輸入幾何必須具有相同的維度。

語法

```
ST_MakeLine(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT、LINESTRING 或 MULTIPOINT。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT、LINESTRING 或 MULTIPOINT。

傳回類型

LINESTRING 子類型的 GEOMETRY。

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 是空點或包含空點，則忽略這些空點。

如果 geom1 和 geom2 為空，則傳回空的 LINESTRING。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom1 和 geom2 具有不同的 SRID 值，則會傳回錯誤。

如果 geom1 或 geom2 不是 POINT、LINESTRING 或 MULTIPOINT，則會傳回錯誤。

如果 geom1 和 geom2 具有不同的維度，則會傳回錯誤。

範例

以下 SQL 會從兩個輸入 linestring 建構 linestring。

```
SELECT ST_MakeLine(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'), ST_GeomFromText('LINESTRING(88.29 39.07,88.42 39.26,88.27
 39.31,88.29 39.07)'));
```

```
st_makeline
-----------
 01020000000800000000C3F5285C8F52534052B81E85EB113D407B14AE47E15A5340C3F5285C8F423D40E17A14AE4751
```

ST_ MakePoint

ST_ 會MakePoint 傳回座標值為輸入值的點幾何圖形。

語法

```
ST_MakePoint(x, y)
```

```
ST_MakePoint(x, y, z)
```

```
ST_MakePoint(x, y, z, m)
```

引數

x

DOUBLE PRECISION 資料類型的值，表示第一個座標。

y

DOUBLE PRECISION 資料類型的值，表示第二個座標。

z

DOUBLE PRECISION 資料類型的值，表示第三個座標。

m

DOUBLE PRECISION 資料類型的值，表示第四個座標。

傳回類型

POINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值會設為 0。

如果 x、y、z 或 m 為 null，則傳回 null。

範例

以下 SQL 會傳回 POINT 子類型的 GEOMETRY 類型，其中包含提供的座標。

```
SELECT ST_AsText(ST_MakePoint(1,3));
```

```
st_astext
-----------
 POINT(1 3)
```

以下 SQL 會傳回 POINT 子類型的 GEOMETRY 類型，其中包含提供的座標。

```
SELECT ST_AsEWKT(ST_MakePoint(1, 2, 3));
```

```
st_asewkt
----------------
 POINT Z (1 2 3)
```

以下 SQL 會傳回 POINT 子類型的 GEOMETRY 類型，其中包含提供的座標。

```
SELECT ST_AsEWKT(ST_MakePoint(1, 2, 3, 4));
```

```
st_asewkt
-------------------
 POINT ZM (1 2 3 4)
```

ST_ MakePolygon

ST_ MakePolygon 有兩個返回多邊形的變體。一個採用單一幾何，另一個採用兩個幾何。

- 第一個變體的輸入是定義輸出多邊形外環的 linestring。
- 第二個變體的輸入是 linestring 和 multilinestring。這兩個都是空的或封閉的。

  輸出多邊形的外環的邊界是輸入 linestring，而多邊形內環的邊界是輸入 multilinestring 中的 linestring。如果輸入 linestring 為空，則會傳回空多邊形。會忽略 multilinestring 中的空 linestring。產生幾何的空間參考系統識別碼 (SRID) 是兩個輸入幾何的公共 SRID。

傳回幾何的維度與輸入幾何的維度相同。外環和內環必須具有相同的維度。

語法

```
ST_MakePolygon(geom1)
```

```
ST_MakePolygon(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。linestring 的值必須是封閉或空的。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 MULTILINESTRING。

傳回類型

POLYGON 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 等於輸入的 SRID。

如果 geom1 或 geom2 為 null，則傳回 null。

如果 geom1 不是線串，則會傳回錯誤。

如果 geom2 不是 multilinestring，則會傳回錯誤。

如果 geom1 未封閉，則會傳回錯誤。

如果 geom1 是單點或未封閉，則會傳回錯誤。

如果 geom2 包含至少一個具有單一點或未封閉的 linestring，則會傳回錯誤。

如果 geom1 和 geom2 具有不同的 SRID 值，則會傳回錯誤。

如果 geom1 和 geom2 具有不同的維度，則會傳回錯誤。

範例

以下 SQL 會從輸入 linestring 傳回多邊形。

```
SELECT ST_AsText(ST_MakePolygon(ST_GeomFromText('LINESTRING(77.29 29.07,77.42
 29.26,77.27 29.31,77.29 29.07)')));
```

```
 st_astext
---------------
POLYGON((77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07))
```

下列 SQL 會從封閉 linestring 和封閉 multilinestring 建立多邊形。Linestring 用於多邊形的外環。Multilinestring 中的 linestring 用於多邊形的內環。

```
SELECT ST_AsEWKT(ST_MakePolygon(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,0 10,0 0)'),
 ST_GeomFromText('MULTILINESTRING((1 1,1 2,2 1,1 1),(3 3,3 4,4 3,3 3))')));
```

```
 st_astext
-----------------------------------
POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 1,1 1),(3 3,3 4,4 3,3 3))
```

ST_ MemSize

ST_ MemSize 返回輸入幾何使用的存儲器空間量（以字節為單位）。此大小取決於幾何的 Amazon Redshift 內部表示法，因此可能會在內部表示法變更時變更。您可以使用此大小，指出 Amazon Redshift 中幾何物件的相對大小。

語法

```
ST_MemSize(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 geom 的固有維度。

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會傳回幾何集合的記憶體大小。

```
SELECT ST_MemSize(ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0
 0)),LINESTRING(20 10,20 0,10 0))')::varchar + ' bytes';
```

```
 ?column?
-----------
```

```
 172 bytes
```

ST_MMax

ST_MMax 會傳回輸入幾何的 m 座標上限。

語法

```
ST_MMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

m 座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DZ 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DM 幾何中 linestring 的最大 m 座標。

```
SELECT ST_MMax(ST_GeomFromEWKT('LINESTRING M (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_mmax
-----------
  8
```

以下 SQL 會傳回 4D 幾何中線串的最大 m 座標。

```
SELECT ST_MMax(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_mmax
-----------
  11
```

ST_MMin

ST_MMin 會傳回輸入幾何的 m 座標下限。

語法

```
ST_MMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

m 座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DZ 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DM 幾何中 linestring 的最小 m 座標。

```
SELECT ST_MMin(ST_GeomFromEWKT('LINESTRING M (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_mmin
-----------
  2
```

以下 SQL 會傳回 4D 幾何中 linestring 的 m 座標下限。

```
SELECT ST_MMin(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_mmin
-----------
  3
```

ST_Multi

ST_Multi 會將幾何轉換為對應的多重類型。如果輸入幾何已經是多重類型或幾何集合，則會傳回其副本。如果輸入幾何是點、linestring 或多邊形，則會分別傳回包含輸入幾何的多點、multilinestring 或多重多邊形。

語法

```
ST_Multi(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

具有子類型 MULTIPOINT、MULTILINESTRING、MULTIPOLYGON 或 GEOMETRYCOLLECTION 的 GEOMETRY。

所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會從輸入多點傳回一個多點。

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('MULTIPOINT((1 2),(3 4))', 4326)));
```

```
    st_asewkt
```

```
-----------------------------------
  SRID=4326;MULTIPOINT((1 2),(3 4))
```

以下 SQL 會從輸入點傳回多點。

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('POINT(1 2)', 4326)));
```

```
    st_asewkt
-----------------------------------
  SRID=4326;MULTIPOINT((1 2))
```

下列 SQL 會從輸入幾何集合傳回幾何集合。

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('GEOMETRYCOLLECTION(POINT(1 2),MULTIPOINT((1
 2),(3 4)))', 4326)));
```

```
    st_asewkt
-----------------------------------
  SRID=4326;GEOMETRYCOLLECTION(POINT(1 2),MULTIPOINT((1 2),(3 4)))
```

ST_NDims

ST_NDims 會傳回幾何的座標維度。ST_NDims 不會考慮幾何的拓撲維度。相反地，它會根據幾何的維度傳回常數值。

語法

```
ST_NDims(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 geom 的固有維度。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回的值	輸入幾何的維度
2	2D
3	3DZ 或 3DM
4	4D

範例

下列 SQL 會傳回 2D linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING(0 0,1 1,2 2,0 0)'));
```

```
st_ndims
-------------
 2
```

以下 SQL 會傳回 3DZ linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING Z(0 0 3,1 1 3,2 2 3,0 0 3)'));
```

```
st_ndims
-------------
 3
```

下列 SQL 會傳回 3DM linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING M(0 0 4,1 1 4,2 2 4,0 0 4)'));
```

```
st_ndims
-------------
```

```
 3
```

下列 SQL 會傳回 4D linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING ZM(0 0 3 4,1 1 3 4,2 2 3 4,0 0 3 4)'));
```

```
st_ndims
-------------
 4
```

ST_NPoints

ST_NPoints 會回輸入幾何或地理中非空點的數量。

語法

```
ST_NPoints(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

INTEGER

如果 geo 是空點，則傳回 0。

如果 geo 為 null，則傳回 null。

範例

以下 SQL 會傳回 linestring 中點的數量。

```
SELECT ST_NPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_npoints
-------------
 4
```

以下 SQL 會傳回地理中 linestring 中點的數量。

```
SELECT ST_NPoints(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
st_npoints
-------------
 4
```

ST_NRings

ST_NRings 會傳回輸入幾何中的環數。

語法

```
ST_NRings(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回的值	幾何子類型
0	在 geom 是 POINT、LINESTRING 、MULTIPOINT 或 MULTILINESTRING 子類型時傳回

傳回的值	幾何子類型
環的數量。	在 geom 是 POLYGON 或 MULTIPOLYGON 子類型時傳回
所有元件中環的數量	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會傳回 multipolygon 中環的數量。

```
SELECT ST_NRings(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((0 0,-10 0,0
 -10,0 0)))'));
```

```
 st_nrings
-------------
 2
```

ST_ NumGeometries

ST_ 會NumGeometries 傳回輸入幾何圖形中的幾何圖形數目。

語法

```
ST_NumGeometries(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 geom 中的幾何數量。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是單一空幾何，則會傳回 0。

如果 geom 是單一非空幾何，則會傳回 1。

如果 geom 是 GEOMETRYCOLLECTION 或 MULTI 子類型，則會傳回幾何的數量。

範例

以下 SQL 會傳回輸入 multilinestring 中的幾何數。

```
SELECT ST_NumGeometries(ST_GeomFromText('MULTILINESTRING((0 0,1 0,0 5),(3 4,13 26))'));
```

```
st_numgeometries
-------------
 2
```

ST_ NumInteriorRings

ST_ 會NumInteriorRings 傳回輸入多邊形幾何圖形中的環數。

語法

```
ST_NumInteriorRings(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER

如果 geom 為 Null，則會傳回 Null。

如果 geom 不是多邊形，則會傳回 Null。

範例

以下 SQL 會傳回輸入多邊形中的內部環數。

```
SELECT ST_NumInteriorRings(ST_GeomFromText('POLYGON((0 0,100 0,100 100,0 100,0 0),(1
 1,1 5,5 1,1 1),(7 7,7 8,8 7,7 7))'));
```

```
 st_numinteriorrings
-------------
 2
```

ST_ NumPoints

ST_ 會NumPoints 傳回輸入幾何圖形中的點數。

語法

```
ST_NumPoints(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER

如果 geom 為 Null，則會傳回 Null。

如果 geom 不是子類型 LINESTRING，則會傳回 Null。

範例

以下 SQL 會傳回輸入 linestring 中點的數量。

```
SELECT ST_NumPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'));
```

```
st_numpoints
```

```
-------------
4
```

下列 SQL 會傳回 Null，因為輸入 geom 不是子類型 LINESTRING。

```
SELECT ST_NumPoints(ST_GeomFromText('MULTIPOINT(1 2,3 4)'));
```

```
st_numpoints
-------------
```

ST_Perimeter

對於輸入面幾何，ST_Perimeter 會傳回 2D 投影的笛卡爾周長 (邊界的長度)。周長單位與表示輸入幾何座標的單位相同。對於點、multipoint 和線形幾何，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中幾何的周長總和。

對於輸入地理，ST_Perimeter 會傳回在球體 (由 SRID 決定) 上所計算輸入面積地理之 2D 投影的測地週長 (邊界長度)。周長的單位是公尺。對於點、multipoint 和線形地理，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中地理的周長總和。

語法

```
ST_Perimeter(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

DOUBLE PRECISION

如果 geo 為 null，則傳回 null。

如果找不到 SRID 值，則會傳回錯誤。

範例

下列 SQL 會傳回 multipolygon 的笛卡爾周長。

```
SELECT ST_Perimeter(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20
 10,10 0)))'));
```

```
 st_perimeter
-------------------------------
    68.2842712474619
```

下列 SQL 會傳回 multipolygon 的笛卡爾周長。

```
SELECT ST_Perimeter(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20
 10,10 0)))'));
```

```
 st_perimeter
-------------------------------
    68.2842712474619
```

以下 SQL 會傳回地理中多邊形的周長。

```
SELECT ST_Perimeter(ST_GeogFromText('SRID=4326;POLYGON((0 0,1 0,0 1,0 0))'));
```

```
 st_perimeter
------------------
 378790.428393693
```

下列 SQL 會傳回地理中 linestring 的周長。

```
SELECT ST_Perimeter(ST_GeogFromText('SRID=4326;LINESTRING(5 0,10 0)'));
```

```
 st_perimeter
--------------
```

```
 0
```

ST_Perimeter2D

ST_Perimeter2D 是 ST_Perimeter 的別名。如需詳細資訊，請參閱 ST_Perimeter。

ST_Point

ST_Point 會傳回輸入座標值中的點幾何。

語法

```
ST_Point(x, y)
```

引數

x

DOUBLE PRECISION 資料類型的值，表示第一個座標。

y

DOUBLE PRECISION 資料類型的值，表示第二個座標。

傳回類型

POINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值會設為 0。

如果 x 或 y 為 Null，則會傳回 Null。

範例

以下 SQL 會從輸入座標建構點幾何。

```
SELECT ST_AsText(ST_Point(5.0, 7.0));
```

```
st_astext
-------------
```

```
POINT(5 7)
```

ST_PointN

ST_PointN 會傳回 linestring 中索引值指定的點。負索引值會從 linestring 結尾向後算起，以便 -1 是最後一個點。

傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_PointN(geom, index)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

索引

資料類型 INTEGER 的值，代表 linestring 中某點的索引。

傳回類型

POINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值會設為 0。

如果 geom 或 index 為 Null，則會傳回 Null。

如果 index 超出範圍，則會傳回 Null。

如果 geom 為空白，則會傳回 Null。

如果 geom 不是 LINESTRING，則會傳回 Null。

範例

下列 SQL 將六點 LINESTRING 的擴充已知文字 (EWKT) 表示法傳回至 GEOMETRY 物件，並傳回 linestring 索引 5 中的點。

```
SELECT ST_AsEWKT(ST_PointN(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5,0
 0)',4326), 5));
```

```
st_asewkt
-------------
 SRID=4326;POINT(0 5)
```

ST_Points

ST_Points 會傳回包含輸入幾何中所有非空點的多點幾何。ST_Points 不會移除輸入中重複的點，包括環形幾何的起點和終點。

語法

```
ST_Points(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

MULTIPOINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值與 geom 相同。

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則傳回空的多點。

範例

下列 SQL 範例會從輸入幾何建構多點幾何。結果為包含輸入幾何中非空點的多點幾何。

```
SELECT ST_AsEWKT(ST_Points(ST_SetSRID(ST_GeomFromText('LINESTRING(1 0,2 0,3 0)'),
 4326)));
```

```
st_asewkt
-------------
SRID=4326;MULTIPOINT((1 0),(2 0),(3 0))
```

```
SELECT ST_AsEWKT(ST_Points(ST_SetSRID(ST_GeomFromText('MULTIPOLYGON(((0 0,1 0,0 1,0
 0)))'), 4326)));
```

```
st_asewkt
-------------
SRID=4326;MULTIPOINT((0 0),(1 0),(0 1),(0 0))
```

ST_Polygon

ST_Polygon 會傳回多邊形幾何，該幾何的外部環是輸入的 linestring，且其值為空間參考系統識別碼 (SRID) 的輸入。

傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_Polygon(linestring, srid)
```

引數

linestring

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是表示 linestring 的 LINESTRING。linestring 的值必須是封閉的。

srid

INTEGER 資料類型的值，表示 SRID。

傳回類型

POLYGON 子類型的 GEOMETRY。

傳回幾何的 SRID 值會設為 srid。

如果 linestring 或 srid 為 Null，則會傳回 Null。

如果 linestring 不是 linestring，則會傳回錯誤。

如果 linestring 並非封閉，則會傳回錯誤。

如果 srid 為負值，則會傳回錯誤。

範例

以下 SQL 會使用 SRID 值建構多邊形。

```
SELECT ST_AsEWKT(ST_Polygon(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'),4356));
```

```
st_asewkt
-------------
 SRID=4356;POLYGON((77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07))
```

ST_ RemovePoint

ST_ 會RemovePoint 傳回已移除索引位置之輸入幾何圖形的點的線串幾何圖形。

索引是以零開始。結果的空間參考系統識別碼 (SRID) 與輸入幾何相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_RemovePoint(geom, index)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

索引

資料類型 INTEGER 的值，代表以零開始的索引位置。

傳回類型

GEOMETRY

如果 geom 或 index 為 Null，則會傳回 Null。

如果 geom 不是子類型 LINESTRING，則會傳回錯誤。

如果 index 超出範圍，則會傳回錯誤。索引位置的有效值介於 0 與 ST_NumPoints(geom) 減 1 之間。

範例

下列 SQL 會移除 linestring 中的最後一個點。

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_RemovePoint(g, ST_NumPoints(g) - 1)) FROM tmp;
```

```
   st_asewkt
-----------------------------------------
 SRID=4326;LINESTRING(0 0,10 0,10 10,5 5)
```

ST_Reverse

ST_Reverse 會反轉線性和面積幾何的頂點順序。對於點或多點幾何，會傳回原始幾何的副本。對於幾何集合，ST_Reverse 會反轉集合中每個幾何的頂點順序。

傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_Reverse(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會反轉 linestring 中點的順序。

```
SELECT ST_AsEWKT(ST_Reverse(ST_GeomFromText('LINESTRING(1 0,2 0,3 0,4 0)', 4326)));
```

```
    st_asewkt
-----------------------------------
  SRID=4326;LINESTRING(4 0,3 0,2 0,1 0)
```

ST_ SetPoint

ST_ SetPoint 返回相對於由索引指定的輸入線串的位置更新坐標的線串。新座標是輸入點的座標。

傳回幾何的維度與 geom1 值的維度相同。如果 geom1 和 geom2 具有不同的維度，則將 geom2 投影到 geom1 的維度。

語法

```
ST_SetPoint(geom1, index, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

索引

INTEGER 資料類型的值，代表索引的位置。0 是指 linestring 左邊的第一個點，1 是指第二點，以此類推。索引可以是負值。-1 是指 linestring 右側的第一個點，-2 是指 linestring 右側的第二個點，以此類推。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

傳回類型

GEOMETRY

如果 geom2 是空點，則傳回 geom1。

如果 geom1、geom2 或 index 為 Null，則會傳回 Null。

如果 geom1 不是線串，則會傳回錯誤。

如果 index 不在有效的索引範圍內，則傳回錯誤。

如果 geom2 不是點，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

範例

下列 SQL 會傳回一個新的 linestring，在這裡我們將輸入 linestring 的第二個點設定為指定點。

```
SELECT ST_AsText(ST_SetPoint(ST_GeomFromText('LINESTRING(1 2, 3 2, 5 2, 1 2)'), 2,
 ST_GeomFromText('POINT(7 9)')));
```

```
st_astext
-------------
 LINESTRING(1 2,3 2,7 9,1 2)
```

下列 SQL 範例會傳回一個新的 linestring，在這裡我們將 linestring 右側的第三個點 (索引為負) 設定為指定點。

```
SELECT ST_AsText(ST_SetPoint(ST_GeomFromText('LINESTRING(1 2, 3 2, 5 2, 1 2)'), -3,
 ST_GeomFromText('POINT(7 9)')));
```

```
st_astext
-------------
```

```
 LINESTRING(1 2,7 9,5 2,1 2)
```

ST_SetSRID

ST_SetSRID 會傳回與輸入幾何相同的幾何，但其會使用空間參考系統識別碼 (SRID) 的值輸入進行更新。

語法

```
ST_SetSRID(geom, srid)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

srid

INTEGER 資料類型的值，表示 SRID。

傳回類型

GEOMETRY

傳回幾何的 SRID 值會設為 srid。

如果 geom 或 srid 為 Null，則會傳回 Null。

如果 srid 為負值，則會傳回錯誤。

範例

以下 SQL 會設定 linestring 的 SRID 值。

```
SELECT ST_AsEWKT(ST_SetSRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
 29.31,77.29 29.07)'),50));
```

```
st_asewkt
-------------
 SRID=50;LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)
```

ST_Simplify

ST_Simplify 會使用具有指定公差的 Ramer-Douglas-Peucker 演算法，傳回輸入幾何的簡化副本。可能不會保留輸入幾何的拓樸。如需演算法的相關資訊，請參閱 Wikipedia 中的 Ramer-Douglas-Peucker 演算法。

當 ST_Simplify 計算距離以簡化幾何時，ST_Simplify 會對輸入幾何的 2D 投影進行操作。

語法

```
ST_Simplify(geom, tolerance)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

tolerance

DOUBLE PRECISION 資料類型的值，代表 Ramer-Douglas-Peucker 演算法的公差等級。如果 tolerance 為負數，則使用零。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

傳回幾何的維度與輸入幾何的維度相同。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 透過 Ramer-Douglas-Peucker 演算法使用歐幾里德距離容差 1 來簡化輸入 linestring。距離的單位與幾何座標的單位相同。

```
SELECT ST_AsEWKT(ST_Simplify(ST_GeomFromText('LINESTRING(0 0,1 2,1 1,2 2,2 1)'), 1));
```

```
 st_asewkt
```

```
-----------
LINESTRING(0 0,1 2,2 1)
```

ST_SRID

ST_SRID 會傳回輸入幾何的空間參考系統識別碼 (SRID)。如需 SRID 的相關資訊，請參閱在 Amazon Redshift 中查詢空間資料。

語法

```
ST_SRID(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER，代表 geom 的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回設定為 SRID 4326 之 linestring 的 SRID 值。

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)',4326));
```

```
st_srid
-------------
 4326
```

下列 SQL 會傳回建構時未設定之 linestring 的 SRID 值。這會導致 SRID 值變成 0。

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_srid
-------------
 0
```

ST_ StartPoint

ST_ StartPoint 返回輸入線串的第一個點。結果的空間參考系統識別碼 (SRID) 值與輸入幾何的 SRID 值相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_StartPoint(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空白，則會傳回 Null。

如果 geom 不是 LINESTRING，則會傳回 Null。

範例

下列 SQL 將四點 LINESTRING 的擴充已知文字 (EWKT) 表示法傳回至 GEOMETRY 物件，並傳回 linestring 的起點。

```
SELECT ST_AsEWKT(ST_StartPoint(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0
 5)',4326)));
```

```
st_asewkt
-------------
```

```
 SRID=4326;POINT(0 0)
```

ST_Touches

ST_Touches 會在兩個輸入幾何的 2D 投影接觸時傳回 true。如果兩個幾何圖形非空白、相交，且沒有共同的內部點，則它們會接觸。

語法

```
ST_Touches(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

下列 SQL 會檢查多邊形是否接觸 linestring。

```
SELECT ST_Touches(ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))'),
 ST_GeomFromText('LINESTRING(20 10,20 0,10 0)'));
```

```
 st_touches
-------------
 t
```

ST_Transform

ST_Transform 會傳回一個新幾何，其座標在由輸入空間參考系統識別碼 (SRID) 定義的空間參考系統中進行轉換。

語法

```
ST_Transform(geom, srid)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

srid

INTEGER 資料類型的值，表示 SRID。

傳回類型

GEOMETRY.

傳回幾何的 SRID 值會設為 srid。

如果 geom 或 srid 為 Null，則會傳回 Null。

如果與輸入 geom 相關聯的 SRID 值不存在，則傳回錯誤。

如果 srid 不存在，則會傳回錯誤。

範例

下列 SQL 會轉換空白幾何集合的 SRID。

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('GEOMETRYCOLLECTION EMPTY', 3857),
 4326));
```

```
             st_asewkt
-----------------------------------
```

```
 SRID=4326;GEOMETRYCOLLECTION EMPTY
```

以下 SQL 會轉換 linestring 的 SRID。

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9,
 -22 -33)', 4326), 26918));
```

```
     st_asewkt
-----------------------------------------------------------------------------------------------
 SRID=26918;LINESTRING(73106.6977300955 15556182.9688576,14347201.5059964
 1545178.32934967,1515090.41262989 9522193.25115316,10491250.83295
 2575457.28410878,5672303.72135968 -5233682.61176205)
```

下列 SQL 會轉換一個多邊形的 SRID。

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('POLYGON Z ((-10 10 -7, -65 10 -6, -10 64
 -5, -10 10 -7), (-11 11 5, -11 12 6, -12 11 7, -11 11 5))', 6989), 6317));
```

```
                                        st_asewkt
-----------------------------------------------------------------------------------------------
 SRID=6317;POLYGON Z ((6186430.2771091 -1090834.57212608
 1100247.33216237,2654831.67853801 -5693304.90741276 1100247.50581055,2760987.41750022
 -486836.575101877 5709710.44137268,6186430.2771091 -1090834.57212608
 1100247.33216237),(6146675.25029258 -1194792.63532103 1209007.1115113,6125027.87562215
 -1190584.81194058 1317403.77865723,6124888.99555252 -1301885.3455052
 1209007.49312929,6146675.25029258 -1194792.63532103 1209007.1115113))
```

ST_Union

ST_Union 會傳回代表兩個幾何聯集的幾何。也就是說，它會合併輸入幾何，以產生沒有重疊的結果幾何。

語法

```
ST_Union(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 或 geom2 為空，則會傳回空的幾何。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合、linestring 或 multilinestring，則會傳回錯誤。

如果 geom1 或 geom2 不是二維 (2D) 幾何，則會傳回錯誤。

範例

下列 SQL 會傳回表示兩個輸入幾何聯集的非空幾何。

```
SELECT ST_AsEWKT(ST_Union(ST_GeomFromText('POLYGON((0 0,100 100,0 200,0 0))'),
 ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))')));
```

```
        st_asewkt
-------------------------
 POLYGON((0 0,0 200,100 100,5 5,10 0,0 0))
```

ST_Within

ST_Within 會在第一個輸入幾何的 2D 投影位於第二個輸入幾何的 2D 投影中時傳回 true。

例如，如果 A 中的每個點都是 B 中的點，且其內部沒有非空白的交集，則幾何 A 便位於幾何 B 中。

ST_Within(A, B) 與 ST_Contains(B, A) 相等。

語法

```
ST_Within(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom2 比較，以判斷其是否位於 geom2 內。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 值不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否位於第二個多邊形中。

```
SELECT ST_Within(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
 ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_within
-----------
 true
```

ST_X

ST_X 會傳回輸入點的第一個座標。

語法

```
ST_X(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

第一個座標的 DOUBLE PRECISION 值。

如果 point 是 Null，則會傳回 Null。

如果 point 是空點，則傳回 null。

如果 point 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回點的第一個座標。

```
SELECT ST_X(ST_Point(1,2));
```

```
st_x
-----------
 1.0
```

ST_XMax

ST_XMax 會傳回輸入幾何的第一個座標上限。

語法

```
ST_XMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第一個座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最大的第一個座標。

```
SELECT ST_XMax(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_xmax
-----------
 77.42
```

ST_XMin

ST_XMin 會傳回輸入幾何的第一個座標下限。

語法

```
ST_XMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第一個座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最小的第一個座標。

```
SELECT ST_XMin(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_xmin
-----------
 77.27
```

ST_Y

ST_Y 會傳回輸入點的第二個座標。

語法

```
ST_Y(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

第二個座標的 DOUBLE PRECISION 值。

如果 point 是 Null，則會傳回 Null。

如果 point 是空點，則傳回 null。

如果 point 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回點的第二個座標。

```
SELECT ST_Y(ST_Point(1,2));
```

```
st_y
-----------
 2.0
```

ST_YMax

ST_YMax 會傳回輸入幾何的第二個座標上限。

語法

```
ST_YMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第二個座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最大的第二個座標。

```
SELECT ST_YMax(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_ymax
-----------
 29.31
```

ST_YMin

ST_YMin 會傳回輸入幾何的第二個座標下限。

語法

```
ST_YMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第二個座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最小的第二個座標。

```
SELECT ST_YMin(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29
 29.07)'));
```

```
st_ymin
-----------
 29.07
```

ST_Z

ST_Z 會傳回輸入點的 z 座標。

語法

```
ST_Z(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

m 座標的 DOUBLE PRECISION 值。

如果 point 是 Null，則會傳回 Null。

如果 point 是 2D 或 3DM 點，則傳回 null。

如果 point 是空點，則傳回 null。

如果 point 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回 3DZ 幾何中點的 z 座標。

```
SELECT ST_Z(ST_GeomFromEWKT('POINT Z (1 2 3)'));
```

```
st_z
-----------
 3
```

以下 SQL 會傳回 4D 幾何中點的 z 座標。

```
SELECT ST_Z(ST_GeomFromEWKT('POINT ZM (1 2 3 4)'));
```

```
st_z
-----------
```

```
 3
```

ST_ZMax

ST_ZMax 會傳回輸入幾何的 z 座標上限。

語法

```
ST_ZMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

z 座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DM 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DZ 幾何中 linestring 的最大 z 座標。

```
SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING Z (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_zmax
-----------
  8
```

以下 SQL 會傳回 4D 幾何中線串的最大 z 座標。

```
SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_zmax
-----------
  10
```

ST_ZMin

ST_ZMin 會傳回輸入幾何的 z 座標下限。

語法

```
ST_ZMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

z 座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DM 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DZ 幾何中 linestring 的最小 z 座標。

```
SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING Z (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_zmin
-----------
  2
```

以下 SQL 會傳回 4D 幾何中 linestring 的 z 座標下限。

```
SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_zmin
-----------
  2
```

SupportsBBox

如果輸入幾何支援使用預先計算的週框方塊進行編碼，SupportsBBox 會傳回 true。如需週框方塊支援的相關資訊，請參閱邊界框。

語法

```
SupportsBBox(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會傳回 true，因為輸入點幾何支援使用週框方塊進行編碼。

```
SELECT SupportsBBox(AddBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
supportsbbox
--------------
```

```
t
```

下列 SQL 會傳回 false，因為輸入點幾何不支援使用週框方塊進行編碼。

```
SELECT SupportsBBox(DropBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
supportsbbox
--------------
f
```

字串函數

主題

- || (串連) 運算子
- ASCII 函數
- BPCHARCMP 函數
- BTRIM 函數
- BTTEXT_PATTERN_CMP 函數
- CHAR_LENGTH 函數
- CHARACTER_LENGTH 函數
- CHARINDEX 函數
- CHR 函數
- COLLATE 函數
- CONCAT 函數
- CRC32 函數
- DIFFERENCE 函數
- INITCAP 函數
- LEFT 和 RIGHT 函數
- LEN 函數
- LENGTH 函數
- LOWER 函數
- LPAD 和 RPAD 函數

- LTRIM 函數
- OCTETINDEX 函數
- OCTET_LENGTH 函數
- POSITION 函數
- QUOTE_IDENT 函數
- QUOTE_LITERAL 函數
- REGEXP_COUNT 函數
- REGEXP_INSTR 函數
- REGEXP_REPLACE 函數
- REGEXP_SUBSTR 函數
- REPEAT 函數
- REPLACE 函數
- REPLICATE 函數
- REVERSE 函數
- RTRIM 函數
- SOUNDEX 函數
- SPLIT_PART 函數
- STRPOS 函數
- STRTOL 函數
- SUBSTRING 函數
- TEXTLEN 函數
- TRANSLATE 函數
- TRIM 函數
- UPPER 函數

字串函數處理和操作字元字串，或評估為字元字串的表達式。當這些函數中的 string 引數是常值時，必須以單引號括住。支援的資料類型包括 CHAR 和 VARCHAR。

下節提供所支援函數的函數名稱、語法及描述。字串的所有偏移都是以一開始。

已取代的僅限領導節點函數

下列字串函數已棄用，因為它們只在領導者節點上執行。如需更多資訊，請參閱僅限領導節點函數

- GET_BYTE
- SET_BIT
- SET_BYTE
- TO_ASCII

|| (串連) 運算子

串連 || 符號兩側的兩個運算式，並傳回串連後的運算式。

類似於 CONCAT 函數。

> **Note**
>
> 如果其中一個或兩個運算式都為 null，則串連的結果為 NULL。

語法

```
expression1 || expression2
```

引數

expression1

CHAR 字串、VARCHAR 字串、二進位運算式或計算結果為其中一種類型的運算式。

expression2

CHAR 字串、VARCHAR 字串、二進位運算式或計算結果為其中一種類型的運算式。

傳回類型

字串的傳回類型與輸入引數的類型相同。例如，串連兩個 VARCHAR 類型的字串會傳回一個 VARCHAR 類型的字串。

範例

下列範例使用 TICKIT 範例資料庫中的 USERS 和 VENUE 表格。如需詳細資訊，請參閱 範本資料庫。

若要串連 USERS 資料表中的 FIRSTNAME 和 LASTNAME 欄位，請使用下列範例。

```
SELECT (firstname || ' ' || lastname) as fullname
FROM users
ORDER BY 1
LIMIT 10;

+-----------------+
|    fullname     |
+-----------------+
| Aaron Banks     |
| Aaron Booth     |
| Aaron Browning  |
| Aaron Burnett   |
| Aaron Casey     |
| Aaron Cash      |
| Aaron Castro    |
| Aaron Dickerson |
| Aaron Dixon     |
| Aaron Dotson    |
+-----------------+
```

若要串連可能包含 Null 的欄，請使用 NVL 和 COALESCE 函數表達式。下列範例使用 NVL，只要遇到 NULL 就傳回 0。

```
SELECT (venuename || ' seats ' || NVL(venueseats, 0)) as seating
FROM venue
WHERE venuestate = 'NV' or venuestate = 'NC'
ORDER BY 1
LIMIT 10;

+-------------------------------------+
|               seating               |
+-------------------------------------+
| Ballys Hotel seats 0                |
| Bank of America Stadium seats 73298 |
| Bellagio Hotel seats 0              |
| Caesars Palace seats 0              |
| Harrahs Hotel seats 0               |
| Hilton Hotel seats 0                |
| Luxor Hotel seats 0                 |
| Mandalay Bay Hotel seats 0          |
| Mirage Hotel seats 0                |
```

```
| New York New York seats 0           |
+-------------------------------------+
```

ASCII 函數

ASCII 函數傳回指定字串中第一個字元的 ASCII 代碼或 Unicode 代碼點。如果字串為空，該函數傳回 0。如果字串為空，它傳回 NULL。

語法

```
ASCII('string')
```

引數

string

CHAR 字串或 VARCHAR 字串。

傳回類型

INTEGER

範例

若要傳回 NULL，請使用下列範例。如果兩個引數相同，則 NULLIF 函數傳回 NULL，因此 ASCII 函數的輸入引數是 NULL。如需詳細資訊，請參閱 NULLIF 函數。

```
SELECT ASCII(NULLIF('',''));

+-------+
| ascii |
+-------+
|  NULL |
+-------+
```

若要傳回 ASCII 碼 0，請使用下列範例。

```
SELECT ASCII('');

+-------+
| ascii |
```

```
+-------+
|     0 |
+-------+
```

若要傳回單字 amazon 的第一個字母的 ASCII 代碼 97，請使用下列範例。

```
SELECT ASCII('amazon');

+-------+
| ascii |
+-------+
|    97 |
+-------+
```

若要傳回單字 Amazon 的第一個字母的 ASCII 代碼 65，請使用下列範例。

```
SELECT ASCII('Amazon');

+-------+
| ascii |
+-------+
|    65 |
+-------+
```

BPCHARCMP 函數

比較兩個字串的值並傳回整數。如果字串相同，則函數會傳回 0。如果第一個字串的字母順序較大，則函數會傳回 1。如果第二個字串較大，則函數會傳回 -1。

如果是多位元組字元，則根據位元組編碼來比較。

BTTEXT_PATTERN_CMP 函數 的同義詞。

語法

```
BPCHARCMP(string1, string2)
```

引數

string1

CHAR 字串或 VARCHAR 字串。

string2

CHAR 字串或 VARCHAR 字串。

傳回類型

INTEGER

範例

下列範例使用 TICKIT 範例資料庫中的 USERS 表格。如需詳細資訊，請參閱 範本資料庫。

若要針對 USERS 資料表中的前十個項目，按字母順序判斷使用者的名字是否大於使用者的姓氏，請使用下列範例。對於 FIRSTNAME 字串的字母順序比 LASTNAME 字串更後面的那些項目，函數會傳回 1。如果 LASTNAME 的字母順序比 FIRSTNAME 更後面，函數會傳回 -1。

```
SELECT userid, firstname, lastname, BPCHARCMP(firstname, lastname)
FROM users
ORDER BY 1, 2, 3, 4
LIMIT 10;

+--------+-----------+-----------+-----------+
| userid | firstname | lastname  | bpcharcmp |
+--------+-----------+-----------+-----------+
|      1 | Rafael    | Taylor    |        -1 |
|      2 | Vladimir  | Humphrey  |         1 |
|      3 | Lars      | Ratliff   |        -1 |
|      4 | Barry     | Roy       |        -1 |
|      5 | Reagan    | Hodge     |         1 |
|      6 | Victor    | Hernandez |         1 |
|      7 | Tamekah   | Juarez    |         1 |
|      8 | Colton    | Roy       |        -1 |
|      9 | Mufutau   | Watkins   |        -1 |
|     10 | Naida     | Calderon  |         1 |
+--------+-----------+-----------+-----------+
```

若要傳回 USERS 資料表中函數傳回 0 的所有項目，請使用下列範例。當 FIRSTNAME 等於 LASTNAME 時，函數傳回 0。

```
SELECT userid, firstname, lastname,
BPCHARCMP(firstname, lastname)
FROM users
```

```
WHERE BPCHARCMP(firstname, lastname)=0
ORDER BY 1, 2, 3, 4;

+--------+-----------+----------+-----------+
| userid | firstname | lastname | bpcharcmp |
+--------+-----------+----------+-----------+
|     62 | Chase     | Chase    |         0 |
|   4008 | Whitney   | Whitney  |         0 |
|  12516 | Graham    | Graham   |         0 |
|  13570 | Harper    | Harper   |         0 |
|  16712 | Cooper    | Cooper   |         0 |
|  18359 | Chase     | Chase    |         0 |
|  27530 | Bradley   | Bradley  |         0 |
|  31204 | Harding   | Harding  |         0 |
+--------+-----------+----------+-----------+
```

BTRIM 函數

BTRIM 函數修剪字串，包括移除開頭和結尾空格，或移除符合選用指定字串的開頭和結尾字元。

語法

```
BTRIM(string [, trim_chars ] )
```

引數

string

要修剪的輸入 VARCHAR 字串。

trim_chars

包含要比對之字元的 VARCHAR 字串。

傳回類型

BTRIM 函數傳回 VARCHAR 字串。

範例

下列範例從字串 ' abc ' 中修剪開頭和結尾空格：

```
select '     abc    ' as untrim, btrim('     abc    ') as trim;
```

```
untrim    | trim
----------+------
   abc    | abc
```

下列範例從字串 'xyzaxyzbxyzcxyz' 中移除開頭和結尾 'xyz' 字串。開頭和結尾的 'xyz' 已移除，但出現在字串內的部分則未移除。

```
select 'xyzaxyzbxyzcxyz' as untrim,
btrim('xyzaxyzbxyzcxyz', 'xyz') as trim;

     untrim      |   trim
-----------------+-----------
 xyzaxyzbxyzcxyz | axyzbxyzc
```

下列範例會從符合 trim _chars 清單 'tes' 中任何字元的字串 'setuphistorycassettes' 中移除開頭和結尾部分。任何出現在輸入字串開頭或結尾的 trim_chars 清單中另一個字元前的 t、e 或 s 都會被移除。

```
SELECT btrim('setuphistorycassettes', 'tes');

     btrim
-----------------
 uphistoryca
```

BTTEXT_PATTERN_CMP 函數

BPCHARCMP 函數的同義詞。

如需詳細資訊，請參閱 BPCHARCMP 函數。

CHAR_LENGTH 函數

LEN 函數的同義詞。

請參閱LEN 函數。

CHARACTER_LENGTH 函數

LEN 函數的同義詞。

請參閱LEN 函數。

CHARINDEX 函數

傳回指定子字串在一個字串內的位置。

如需相似函數，請參閱 POSITION 函數 和 STRPOS 函數。

語法

```
CHARINDEX( substring, string )
```

引數

substring

在 string 內要搜尋的子字串。

string

要搜尋的字串或欄。

傳回類型

INTEGER

CHARINDEX 函數傳回對應於子字串位置的 INTEGER (以 1 開始，不是以零開始)。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。如果在字串內找不到子字串，CHARINDEX 會傳回 0。

範例

若要顯示字串 fish 在單字 dog 內的位置，請使用下列範例。

```
SELECT CHARINDEX('fish', 'dog');

+-----------+
| charindex |
+-----------+
|         0 |
```

```
+-----------+
```

若要顯示字串 fish 在單字 dogfish 內的位置，請使用下列範例。

```
SELECT CHARINDEX('fish', 'dogfish');

+-----------+
| charindex |
+-----------+
|         4 |
+-----------+
```

以下範例使用 TICKIT 範例資料庫中的 SALES 資料表。如需詳細資訊，請參閱 範本資料庫。

若要從 SALES 資料表中傳回佣金超過 999.00 的不同銷售交易次數，請使用下列範例。此命令透過檢查佣金值開頭的小數是否超過 4 位元來計算大於 999.00 的佣金。

```
SELECT DISTINCT CHARINDEX('.', commission), COUNT (CHARINDEX('.', commission))
FROM sales
WHERE CHARINDEX('.', commission) > 4
GROUP BY CHARINDEX('.', commission)
ORDER BY 1,2;

+-----------+-------+
| charindex | count |
+-----------+-------+
|         5 |   629 |
+-----------+-------+
```

CHR 函數

CHR 函數傳回符合輸入參數指定之 ASCII 字碼指標值的字元。

語法

```
CHR(number)
```

引數

number

輸入參數是代表 ASCII 字碼指標值的 INTEGER。

傳回類型

CHAR

如果 ASCII 字元符合輸入值，CHR 函數會傳回 CHAR 字串。如果輸入數字沒有相符的 ASCII，函數會傳回 NULL。

範例

若要傳回與 ASCII 碼點 0 對應的字元，請使用下列範例。請注意，CHR 函數會針對輸入 0 傳回 NULL。

```
SELECT CHR(0);

+-----+
| chr |
+-----+
|     |
+-----+
```

若要傳回與 ASCII 碼點 65 對應的字元，請使用下列範例。

```
SELECT CHR(65);

+-----+
| chr |
+-----+
| A   |
+-----+
```

若要傳回以大寫 A (ASCII 字碼指標 65) 開頭的活動名稱，請使用下列範例。下列範例使用 TICKIT 範例資料庫中的 EVENT 資料表。如需詳細資訊，請參閱 範本資料庫。

```
SELECT DISTINCT eventname FROM event
WHERE SUBSTRING(eventname, 1, 1)=CHR(65) LIMIT 5;

+-----------------------+
|       eventname       |
+-----------------------+
| A Catered Affair      |
```

```
| As You Like It        |
| A Man For All Seasons |
| Alan Jackson          |
| Armando Manzanero     |
+-----------------------+
```

COLLATE 函數

COLLATE 函數會覆寫字串欄或運算式的定序。

如需如何使用資料庫定序建立資料表的資訊，請參閱CREATE TABLE。

如需如何使用資料庫定序建立資料庫的資訊，請參閱CREATE DATABASE。

語法

```
COLLATE( string, 'case_sensitive' | 'case_insensitive');
```

引數

string

您要覆寫的字串欄或運算式。

'case_sensitive' | 'case_insensitive'

定序名稱的字串常數。Amazon Redshift 只支援 case_sensitive 或 case_insensitive。

傳回類型

COLLATE 函數傳回 VARCHAR 或 CHAR，取決於第一個輸入運算式類型。此函數只會變更第一個輸入引數的定序，而不會變更其輸出值。

範例

若要建立資料表 T 並將資料表 T 中的 col1 定義為 case_sensitive，請使用下列範例。

```
CREATE TABLE T ( col1 Varchar(20) COLLATE case_sensitive );

INSERT INTO T VALUES ('john'),('JOHN');
```

當您執行第一個查詢時，Amazon Redshift 只會傳回 john。在 col1 上執行 COLLATE 函數之後，定序會變成 case_insensitive。第二個查詢會傳回 john 和 JOHN。

```
SELECT * FROM T WHERE col1 = 'john';

+------+
| col1 |
+------+
| john |
+------+

SELECT * FROM T WHERE COLLATE(col1, 'case_insensitive') = 'john';

+------+
| col1 |
+------+
| john |
| JOHN |
+------+
```

若要建立資料表 A 並將資料表 A 中的 col1 定義為 case_insensitive，請使用下列範例。

```
CREATE TABLE A ( col1 Varchar(20) COLLATE case_insensitive );

INSERT INTO A VALUES ('john'),('JOHN');
```

當您執行第一個查詢時，Amazon Redshift 會同時傳回 john 和 JOHN。在 col1 上執行 COLLATE 函數之後，定序會變成 case_sensitive。第二個查詢只會傳回 john。

```
SELECT * FROM A WHERE col1 = 'john';

+------+
| col1 |
+------+
| john |
| JOHN |
+------+

SELECT * FROM A WHERE COLLATE(col1, 'case_sensitive') = 'john';

+------+
| col1 |
```

```
+------+
| john |
+------+
```

CONCAT 函數

CONCAT 函數會串連兩個運算式，並傳回產生的運算式。若要串連兩個以上的運算式，請使用巢狀 CONCAT 函數。兩個運算式之間的串連運算子 (||) 產生與 CONCAT 函數相同的結果。

語法

```
CONCAT ( expression1, expression2 )
```

引數

expression1、expression2

這兩個引數都可以是固定長度的字元字串、可變長度字串、二進位運算式或計算結果為這些輸入之一的運算式。

傳回類型

CONCAT 傳回一個運算式。運算式的資料類型與輸入引數的類型相同。

如果輸入運算式的類型不同，Amazon Redshift 會嘗試以隱含方式輸入轉換其中一個運算式。如果無法轉換數值，系統會傳回一個錯誤。

使用須知

- 對於 CONCAT 函數和串連運算子，如果一個或兩個運算式為 Null，則串連的結果為 Null。

範例

下列範例串連兩個字元常值：

```
SELECT CONCAT('December 25, ', '2008');

concat
-------------------
December 25, 2008
```

```
(1 row)
```

下列查詢 (使用 || 運算子，而不是 CONCAT) 產生相同的結果：

```
SELECT 'December 25, '||'2008';

?column?
-------------------
December 25, 2008
(1 row)
```

下列範例會使用另一個 CONCAT 函數內的巢狀 CONCAT 函數來連接三個字元字串：

```
SELECT CONCAT('Thursday, ', CONCAT('December 25, ', '2008'));

concat
-----------------------------
Thursday, December 25, 2008
(1 row)
```

若要串連可能包含 NULL 的欄，請使用 NVL 和 COALESCE 函數，當遇到 NULL 時會傳回指定值。下列範例使用 NVL，只要遇到 NULL 就傳回 0。

```
SELECT CONCAT(venuename, CONCAT(' seats ', NVL(venueseats, 0))) AS seating
FROM venue WHERE venuestate = 'NV' OR venuestate = 'NC'
ORDER BY 1
LIMIT 5;

seating
-----------------------------------
Ballys Hotel seats 0
Bank of America Stadium seats 73298
Bellagio Hotel seats 0
Caesars Palace seats 0
Harrahs Hotel seats 0
(5 rows)
```

下列查詢串連 VENUE 資料表中的 CITY 和 STATE 值：

```
SELECT CONCAT(venuecity, venuestate)
FROM venue
```

```
WHERE venueseats > 75000
ORDER BY venueseats;

concat
-------------------
DenverCO
Kansas CityMO
East RutherfordNJ
LandoverMD
(4 rows)
```

下列查詢使用巢狀 CONCAT 函數。此查詢串連 VENUE 資料表中的 CITY 和 STATE 值，但以逗號和空格來分隔產生的字串：

```
SELECT CONCAT(CONCAT(venuecity,', '),venuestate)
FROM venue
WHERE venueseats > 75000
ORDER BY venueseats;

concat
----------------------
Denver, CO
Kansas City, MO
East Rutherford, NJ
Landover, MD
(4 rows)
```

下列範例會串連兩個二進位運算式。其中 abc 是二進位值 (以十六進位表示 616263)，且 def 是二進位值 (使用十六進位表示 646566)。結果會自動顯示為二進位值的十六進位表示。

```
SELECT CONCAT('abc'::VARBYTE, 'def'::VARBYTE);

concat
-------------------
616263646566
```

CRC32 函數

CRC32 是用於錯誤檢測的函數。函數使用 CRC32 演算法來偵測來源和目標資料之間的變更。CRC32 函數將可變長度字串轉換為 8 個字元的字串，此為 32 位元二進位序列之十六進位值的文字表示法。若要偵測來源與目標資料之間的變更，請在來源資料上使用 CRC32 函數並儲存輸出。然後，在目標資料

上使用 CRC32 函數，並將該輸出與來源資料的輸出進行比較。如果資料沒有被修改，輸出將是相同的，如果資料被修改，輸出將不同。

語法

```
CRC32(string)
```

引數

string

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

CRC32 函數傳回 8 個字元的字串，此為 32 位元二進位序列之十六進位值的文字表示法。Amazon Redshift CRC32 函數以 CRC-32C 多項式為基礎。

範例

為了顯示字串 Amazon Redshift 的 8 位值。

```
SELECT CRC32('Amazon Redshift');

+----------+
|  crc32   |
+----------+
| f2726906 |
+----------+
```

DIFFERENCE 函數

DIFFERENCE 函數比較兩個字串的 American Soundex 代碼。該函數傳回 INTEGER，以指示 Soundex 代碼之間的比對字元的數量。

一個 Soundex 代碼是一個四個字元長的字串。Soundex 代碼代表單詞的發音方式，而不是拼寫方式。例如，Smith 和 Smyth 具有相同的 Soundex 代碼。

語法

```
DIFFERENCE(string1, string2)
```

引數

string1

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

string2

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

INTEGER

DIFFERENCE 函數傳回 0-4 之間的 INTEGER 值，用於計算兩個字串的 American Soundex 代碼中比對字元的數量。Soundex 代碼有 4 個字元，因此當字串的 American Soundex 代碼值的所有 4 個字元都相同時，DIFFERENCE 函數會傳回 4。如果兩個字串中的一個為空，則 DIFFERENCE 傳回 0。如果字串都不包含有效字元，該函數傳回 1。DIFFERENCE 函數只會轉換英文字母小寫或大寫 ASCII 字元，包括 a-z 和 A-Z。DIFFERENCE 會忽略其他字元。

範例

若要比較字串 % 和 @ 的 Soundex 值，請使用下列範例。該函數傳回 1，因為字串都不包含有效的字元。

```
SELECT DIFFERENCE('%', '@');

+------------+
| difference |
+------------+
|          1 |
+------------+
```

若要比較 Amazon 的 Soundex 值和一個空字串，請使用下列範例。該函數傳回 0，因為兩個字串中的一個是空的。

```
SELECT DIFFERENCE('Amazon', '');

+------------+
| difference |
+------------+
```

```
|          0 |
+------------+
```

若要比較字串 Amazon 和 Ama 的 Soundex 值，請使用下列範例。該函數傳回 2，因為字串的 Soundex 值的 2 個字元是相同的。

```
SELECT DIFFERENCE('Amazon', 'Ama');

+------------+
| difference |
+------------+
|          2 |
+------------+
```

若要比較字串 Amazon 和 +-*/%Amazon 的 Soundex 值，請使用下列範例。該函數傳回 4，因為字串的 Soundex 值的所有 4 個字元都是相同的。請注意，函數會忽略第二個字串中的無效字元 +-*/%。

```
SELECT DIFFERENCE('Amazon', '+-*/%Amazon');

+------------+
| difference |
+------------+
|          4 |
+------------+
```

若要比較字串 AC/DC 和 Ay See Dee See 的 Soundex 值，請使用下列範例。函數傳回 4，因為字串的 Soundex 值的所有 4 個字元都相同。

```
SELECT DIFFERENCE('AC/DC', 'Ay See Dee See');

+------------+
| difference |
+------------+
|          4 |
+------------+
```

INITCAP 函數

將指定字串中每個單字的第一個字母變成大寫。INITCAP 支援 UTF-8 多位元組字元，每個字元最多 4 個位元組。

語法

```
INITCAP(string)
```

引數

string

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

VARCHAR

使用須知

INITCAP 函數將字串中每個單字的第一個字母變成大寫，而後續任何字母都變成 (或保持) 小寫。因此，必須了解哪些字元 (除了空白字元) 做為單字分隔符號。單字分隔符號字元是任何非英數字元，包括標點符號、記號及控制字元。以下所有字元都是單字分隔符號：

```
! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~
```

Tab、新行字元、換頁、換行及歸位也是單字分隔符號。

範例

下列範例會使用 TICKIT 範例資料庫中 CATEGORY 和 USERS 表格中的資料。如需詳細資訊，請參閱範本資料庫。

若要將 CATDESC 欄中每個單字的首字母變成大寫，請使用下列範例。

```
SELECT catid, catdesc, INITCAP(catdesc)
FROM category
ORDER BY 1, 2, 3;

+-------+--------------------------------------------
+--------------------------------------------+
| catid |                  catdesc                   |                 initcap
          |
+-------+--------------------------------------------
+--------------------------------------------+
```

```
|     1 | Major League Baseball                        | Major League Baseball
          |
|     2 | National Hockey League                       | National Hockey League
          |
|     3 | National Football League                     | National Football League
          |
|     4 | National Basketball Association              | National Basketball Association
          |
|     5 | Major League Soccer                          | Major League Soccer
          |
|     6 | Musical theatre                              | Musical Theatre
          |
|     7 | All non-musical theatre                      | All Non-Musical Theatre
          |
|     8 | All opera and light opera                    | All Opera And Light Opera
          |
|     9 | All rock and pop music concerts              | All Rock And Pop Music Concerts
          |
|    10 | All jazz singers and bands                   | All Jazz Singers And Bands
          |
|    11 | All symphony, concerto, and choir concerts | All Symphony, Concerto, And
 Choir Concerts |
+-------+----------------------------------------------
+-----------------------------------------+
```

若要顯示 INITCAP 函數不保留非單字開頭的大寫字元，請使用下列範例。例如，字串 MLB 會變成 Mlb。

```
SELECT INITCAP(catname)
FROM category
ORDER BY catname;

+-----------+
|  initcap  |
+-----------+
| Classical |
| Jazz      |
| Mlb       |
| Mls       |
| Musicals  |
| Nba       |
| Nfl       |
| Nhl       |
```

```
| Opera     |
| Plays     |
| Pop       |
+-----------+
```

若要將空格以外的非英數字元顯示為文字分隔符號，請使用下列範例。每個字串中的幾個字母將被大寫。

```
SELECT email, INITCAP(email)
FROM users
ORDER BY userid DESC LIMIT 5;

+-----------------------------------+-----------------------------------+
|               email               |              initcap              |
+-----------------------------------+-----------------------------------+
| urna.Ut@egetdictumplacerat.edu    | Urna.Ut@Egetdictumplacerat.Edu    |
| nibh.enim@egestas.ca              | Nibh.Enim@Egestas.Ca              |
| in@Donecat.ca                     | In@Donecat.Ca                     |
| sodales@blanditviverraDonec.ca    | Sodales@Blanditviverradonec.Ca    |
| sociis.natoque.penatibus@vitae.org | Sociis.Natoque.Penatibus@Vitae.Org |
+-----------------------------------+-----------------------------------+
```

LEFT 和 RIGHT 函數

這些函數從字元字串最左邊或最右邊傳回指定的字元數。

數目以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。

語法

```
LEFT( string,  integer )

RIGHT( string,  integer )
```

引數

string

CHAR 字串、VARCHAR 字串或任何計算結果為 CHAR 或 VARCHAR 字串的運算式。

integer

正整數。

傳回類型

VARCHAR

範例

下列範例會使用 TICKIT 範例資料庫中 EVENT 資料表的資料。如需詳細資訊，請參閱 範本資料庫。

若要從 ID 介於 1000 和 1005 之間的活動名稱中，傳回最左邊 5 個和最右邊 5 個字元，請使用下列範例。

```
SELECT eventid, eventname,
LEFT(eventname,5) AS left_5,
RIGHT(eventname,5) AS right_5
FROM event
WHERE eventid BETWEEN 1000 AND 1005
ORDER BY 1;

+---------+----------------+--------+---------+
| eventid |   eventname    | left_5 | right_5 |
+---------+----------------+--------+---------+
|    1000 | Gypsy          | Gypsy  | Gypsy   |
|    1001 | Chicago        | Chica  | icago   |
|    1002 | The King and I | The K  | and I   |
|    1003 | Pal Joey       | Pal J  | Joey    |
|    1004 | Grease         | Greas  | rease   |
|    1005 | Chicago        | Chica  | icago   |
+---------+----------------+--------+---------+
```

LEN 函數

以字元數傳回指定字串的長度。

語法

LEN 是 LENGTH 函數、CHAR_LENGTH 函數、CHARACTER_LENGTH 函數 及 TEXTLEN 函數 的同義詞。

```
LEN(expression)
```

引數

運算式

CHAR 字串、VARCHAR 字串、VARBYTE 運算式或隱含評估為 CHAR、VARCHAR 或 VARBYTE 類型的運算式。

傳回類型

INTEGER

LEN 函數傳回整數，表示輸入字串中的字元數。

如果輸入字串是字串，則 LEN 函數會傳回多位元組字串中的實際字元數，而不是位元組數。例如，需要 VARCHAR(12) 欄來儲存三個四位元組中文字元。LEN 函數針對此相同字串傳回 3。若要取得字串的長度 (以位元組為單位)，請使用 OCTET_LENGTH 函數。

使用須知

如果 expression 是 CHAR 字串，不計算結尾空格。

如果 expression 是 VARCHAR 字串，則計算結尾空格。

範例

若要傳回字串 `français` 中的位元組數和字元數，請使用下列範例。

```
SELECT OCTET_LENGTH('français'),
LEN('français');

+--------------+-----+
| octet_length | len |
+--------------+-----+
|            9 |   8 |
+--------------+-----+
```

若要在不使用 OCTET_LENGTH 函數的情況下傳回字串 `français` 中的位元組數和字元數，請使用下列範例。如需更多資訊，請參閱CAST 函數。

```
SELECT LEN(CAST('français' AS VARBYTE)) as bytes, LEN('français');
```

```
+-------+-----+
| bytes | len |
+-------+-----+
|     9 |   8 |
+-------+-----+
```

若要傳回沒有結尾空格的字串 cat 中的字元數、具有三個結尾空格的 cat 中的字元數、將具有三個結尾空格的 cat 轉換為長度為 6 的 CHAR，以及將具有三個結尾空格的 cat 轉換為長度為 6 的 VARCHAR，請使用以下範例。請注意，該函數不會計算 CHAR 字串的結尾空格，但會計算 VARCHAR 字串的結尾空格。

```
SELECT LEN('cat'), LEN('cat   '), LEN(CAST('cat   ' AS CHAR(6))) AS len_char,
 LEN(CAST('cat   ' AS VARCHAR(6))) AS len_varchar;

+-----+-----+----------+-------------+
| len | len | len_char | len_varchar |
+-----+-----+----------+-------------+
|   3 |   6 |        3 |           6 |
+-----+-----+----------+-------------+
```

下列範例使用 TICKIT 範例資料庫中 VENUE 表格中的資料。如需詳細資訊，請參閱 範本資料庫。

若要傳回 VENUE 資料表中前十個最長的會場名稱，請使用下列範例。

```
SELECT venuename, LEN(venuename)
FROM venue
ORDER BY 2 DESC, 1
LIMIT 10;

+----------------------------------------+-----+
|               venuename                | len |
+----------------------------------------+-----+
| Saratoga Springs Performing Arts Center |  39 |
| Lincoln Center for the Performing Arts  |  38 |
| Nassau Veterans Memorial Coliseum       |  33 |
| Jacksonville Municipal Stadium          |  30 |
| Rangers BallPark in Arlington           |  29 |
| University of Phoenix Stadium           |  29 |
| Circle in the Square Theatre            |  28 |
| Hubert H. Humphrey Metrodome            |  28 |
| Oriole Park at Camden Yards             |  27 |
| Dick's Sporting Goods Park              |  26 |
```

```
+----------------------------------------+-----+
```

LENGTH 函數

LEN 函數的同義詞。

請參閱LEN 函數。

LOWER 函數

將字串轉換成小寫。LOWER 支援 UTF-8 多位元組字元，每個字元最多 4 個位元組。

語法

```
LOWER(string)
```

引數

string

評估為 VARCHAR 類型的 VARCHAR 字串或運算式。

傳回類型

string

LOWER 函數傳回與輸入字串的資料類型相同的字串。例如，如果輸入是 CHAR 字串，該函數將傳回 CHAR 字串。

範例

下列範例會使用 TICKIT 範例資料庫中 CATEGORY 表格中的資料。如需詳細資訊，請參閱 範本資料庫。

若要將 CATNAME 欄中的 VARCHAR 字串轉換為小寫，請使用下列範例。

```
SELECT catname, LOWER(catname) FROM category ORDER BY 1,2;

+-----------+-----------+
|  catname  |   lower   |
+-----------+-----------+
```

```
| Classical | classical |
| Jazz      | jazz      |
| MLB       | mlb       |
| MLS       | mls       |
| Musicals  | musicals  |
| NBA       | nba       |
| NFL       | nfl       |
| NHL       | nhl       |
| Opera     | opera     |
| Plays     | plays     |
| Pop       | pop       |
+-----------+-----------+
```

LPAD 和 RPAD 函數

這些函數根據指定的長度，將字元附加到字串的前面或後面。

語法

```
LPAD(string1, length, [ string2 ])
```

```
RPAD(string1, length, [ string2 ])
```

引數

string1

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

長度

整數，定義函數結果的長度。字串長度以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。如果 string1 比指定的長度更長，則會截斷 (從右邊)。如果 length 是零或負數，則函數結果為空字串。

string2

(選用) 附加到 string1 前面或後面的一或多個字元。如果未指定此引數，則使用空格。

傳回類型

VARCHAR

範例

下列範例會使用 TICKIT 範例資料庫中 EVENT 資料表中的資料。如需詳細資訊，請參閱 範本資料庫。

若要將一組指定的活動名稱截斷至 20 個字元，並在較短名稱的前面附加空格，請使用下列範例。

```
SELECT LPAD(eventname, 20) FROM event
WHERE eventid BETWEEN 1 AND 5 ORDER BY 1;

+----------------------+
|         lpad         |
+----------------------+
|               Salome |
|        Il Trovatore |
|       Boris Godunov |
|     Gotterdammerung |
|La Cenerentola (Cind |
+----------------------+
```

若要將同一組活動名稱截斷至 20 個字元，但在較短名稱的後面附加 0123456789，請使用下列範例。

```
SELECT RPAD(eventname, 20,'0123456789') FROM event
WHERE eventid BETWEEN 1 AND 5 ORDER BY 1;

+----------------------+
|         rpad         |
+----------------------+
| Boris Godunov0123456 |
| Gotterdammerung01234 |
| Il Trovatore01234567 |
| La Cenerentola (Cind |
| Salome01234567890123 |
+----------------------+
```

LTRIM 函數

從字串開頭修剪字元。刪除只包含在修剪字元清單中的字元的最長字串。當修剪字元沒有出現在輸入字串中時，就會完成修剪。

語法

```
LTRIM( string [, trim_chars] )
```

引數

string

要修剪的字串資料行、運算式或字串常值。

trim_chars

字串欄、運算式或字串常值，代表要從 string 開頭修剪的字元。如果未指定，則會使用空格作為修剪字元。

傳回類型

LTRIM 函數會傳回與輸入 string 的資料類型相同的字元字串 (CHAR 或 VARCHAR)。

範例

下列範例從 listime 欄中擷取年份。字串常值 '2008-' 中的修剪字元表示要從左側修剪的字元。如果使用修剪字元 '028-'，則可以得到相同的結果。

```
select listid, listtime, ltrim(listtime, '2008-')
from listing
order by 1, 2, 3
limit 10;

listid |      listtime       |     ltrim
-------+---------------------+----------------
     1 | 2008-01-24 06:43:29 | 1-24 06:43:29
     2 | 2008-03-05 12:25:29 | 3-05 12:25:29
     3 | 2008-11-01 07:35:33 | 11-01 07:35:33
     4 | 2008-05-24 01:18:37 | 5-24 01:18:37
     5 | 2008-05-17 02:29:11 | 5-17 02:29:11
     6 | 2008-08-15 02:08:13 | 15 02:08:13
     7 | 2008-11-15 09:38:15 | 11-15 09:38:15
     8 | 2008-11-09 05:07:30 | 11-09 05:07:30
     9 | 2008-09-09 08:03:36 | 9-09 08:03:36
    10 | 2008-06-17 09:44:54 | 6-17 09:44:54
```

當 trim_chars 中任何字元出現在 string 開頭時，LTRIM 會移除這些字元。下列範例修剪 VENUENAME (這是 VARCHAR 欄) 開頭出現的 'C'、'D' 和 'G' 字元。

```
select venueid, venuename, ltrim(venuename, 'CDG')
```

```
from venue
where venuename like '%Park'
order by 2
limit 7;

venueid | venuename                  | btrim
--------+----------------------------+-------------------------
    121 | ATT Park                   | ATT Park
    109 | Citizens Bank Park         | itizens Bank Park
    102 | Comerica Park              | omerica Park
      9 | Dick's Sporting Goods Park | ick's Sporting Goods Park
     97 | Fenway Park                | Fenway Park
    112 | Great American Ball Park   | reat American Ball Park
    114 | Miller Park                | Miller Park
```

下列範例使用修剪字元 2，這是從 venueid 欄擷取的。

```
select ltrim('2008-01-24 06:43:29', venueid)
from venue where venueid=2;

ltrim
------------------
008-01-24 06:43:29
```

下列範例不會修剪任何字元，因為在 '0' 修剪字元之前找到 2。

```
select ltrim('2008-01-24 06:43:29', '0');

ltrim
-------------------
2008-01-24 06:43:29
```

下列範例會使用預設的空格修剪字元，並從字串的開頭修剪兩個空格。

```
select ltrim('  2008-01-24 06:43:29');

ltrim
-------------------
2008-01-24 06:43:29
```

OCTETINDEX 函數

所述 OCTETINDEX 函數傳回一個字串作為一個位元組數中的子字串的位置。

語法

```
OCTETINDEX(substring, string)
```

引數

substring

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

string

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

INTEGER

OCTETINDEX 函數會傳回與 string 中 substring 位置相對應的 INTEGER 值，做為位元組數目，其中 string 中的第一個字元會計為 1。如果 string 不包含多位元組字元，結果會等於 CHARINDEX 函數的結果。如果 string 不包含 substring，則函數會傳回 0。如果 substring 為空，則函數會傳回 1。

範例

若要傳回字串 Amazon Redshift 中子字串 q 的位置，請使用下列範例。此範例會傳回 0，因為 substring 不在 string 中。

```
SELECT OCTETINDEX('q', 'Amazon Redshift');

+------------+
| octetindex |
+------------+
|          0 |
+------------+
```

若要傳回字串 Amazon Redshift 中空白子字串的位置，請使用下列範例。這個範例會傳回 1，因為 substring 是空的。

```
SELECT OCTETINDEX('', 'Amazon Redshift');

+------------+
| octetindex |
+------------+
|          1 |
+------------+
```

若要傳回字串 Amazon Redshift 中子字串 Redshift 的位置，請使用下列範例。這個範例會傳回 8，因為 substring 是從 string 的第八個位元組開始。

```
SELECT OCTETINDEX('Redshift', 'Amazon Redshift');

+------------+
| octetindex |
+------------+
|          8 |
+------------+
```

若要傳回字串 Amazon Redshift 中子字串 Redshift 的位置，請使用下列範例。這個範例會傳回 21，因為 string 的前六個字元是雙位元組字元。

```
SELECT OCTETINDEX('Redshift', 'Άμαζον Amazon Redshift');

+------------+
| octetindex |
+------------+
|         21 |
+------------+
```

OCTET_LENGTH 函數

傳回指定字串的長度 (以位元組為單位)。

語法

```
OCTET_LENGTH(expression)
```

引數

運算式

CHAR 字串、VARCHAR 字串、VARBYTE 運算式或隱含評估為 CHAR、VARCHAR 或 VARBYTE 類型的運算式。

傳回類型

INTEGER

OCTET_LENGTH 函數傳回整數，表示輸入字串中的位元組數。

如果輸入字串是字元字串，則 LEN 函數會傳回多位元組字串中的實際字元數，而不是位元組數。例如，需要 VARCHAR(12) 欄來儲存三個四位元組中文字元。OCTET_LENGTH 函數將針對該字串傳回 12，而 LEN 函數會針對相同的字串傳回 3。

使用須知

如果 expression 是 CHAR 字串，該函數傳回 CHAR 字串的長度。例如，CHAR(6) 輸入的輸出是 CHAR(6)。

如果 expression 是 VARCHAR 字串，則計算結尾空格。

範例

若要在具有三個結尾空格的字串 francais 轉換為 CHAR 和 VARCHAR 類型時傳回位元組數，請使用下列範例。如需更多資訊，請參閱CAST 函數。

```
SELECT OCTET_LENGTH(CAST('francais   ' AS CHAR(15))) AS octet_length_char,
 OCTET_LENGTH(CAST('francais   ' AS VARCHAR(15))) AS octet_length_varchar;

+------------------+----------------------+
| octet_length_char | octet_length_varchar |
+------------------+----------------------+
|                15 |                   11 |
+------------------+----------------------+
```

若要傳回字串 français 中的位元組數和字元數，請使用下列範例。

```
SELECT OCTET_LENGTH('français'), LEN('français');
```

```
+--------------+-----+
| octet_length | len |
+--------------+-----+
|            9 |   8 |
+--------------+-----+
```

若要在字串 `français` 轉換為 VARBYTE 時傳回位元組數，請使用下列範例。

```
SELECT OCTET_LENGTH(CAST('français' AS VARBYTE));

+--------------+
| octet_length |
+--------------+
|            9 |
+--------------+
```

POSITION 函數

傳回指定子字串在一個字串內的位置。

如需相似函數，請參閱 CHARINDEX 函數 和 STRPOS 函數。

語法

```
POSITION(substring IN string )
```

引數

substring

在 string 內要搜尋的子字串。

string

要搜尋的字串或欄。

傳回類型

POSITION 函數傳回對應於子字串位置的 INTEGER (以 1 開始，不是以零開始)。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。如果在字串內找不到子字串，POSITION 會傳回 0。

範例

若要顯示字串 fish 在單字 dog 內的位置，請使用下列範例。

```
SELECT POSITION('fish' IN 'dog');

+-----------+
|  position |
+-----------+
|         0 |
+-----------+
```

若要顯示字串 fish 在單字 dogfish 內的位置，請使用下列範例。

```
SELECT POSITION('fish' IN 'dogfish');

+-----------+
|  position |
+-----------+
|         4 |
+-----------+
```

以下範例使用 TICKIT 範例資料庫中的 SALES 資料表。如需詳細資訊，請參閱 範本資料庫。

若要從 SALES 資料表中傳回佣金超過 999.00 的不同銷售交易次數，請使用下列範例。此命令透過檢查佣金值開頭的小數是否超過 4 位元來計算大於 999.00 的佣金。

```
SELECT DISTINCT POSITION('.' IN commission), COUNT (POSITION('.' IN commission))
FROM sales
WHERE POSITION('.' IN commission) > 4
GROUP BY POSITION('.' IN commission)
ORDER BY 1,2;

+-----------+-------+
|  position | count |
+-----------+-------+
|         5 |   629 |
+-----------+-------+
```

QUOTE_IDENT 函數

QUOTE_IDENT 函數或會將指定的字串傳回為帶有開頭雙引號和結尾雙引號的字串。函數輸出可以用作 SQL 陳述式中的識別碼。該函數適當地加倍任何嵌入的雙引號。

只有在必須建立有效識別碼時、當字串包含非識別碼字元時，或否則會變為小寫時，QUOTE_IDENT 才會增加雙引號。若要一律傳回以單引號括住的字串，請使用 QUOTE_LITERAL。

語法

```
QUOTE_IDENT(string)
```

引數

string

CHAR 或 VARCHAR 字串。

傳回類型

QUOTE_IDENT 函數傳回與輸入字串相同類型的字串。

範例

若要傳回含雙引號的字串 "CAT"，請使用下列範例。

```
SELECT QUOTE_IDENT('"CAT"');

+-------------+
| quote_ident |
+-------------+
| """CAT"""   |
+-------------+
```

下列範例會使用 TICKIT 範例資料庫中 CATEGORY 表格中的資料。如需詳細資訊，請參閱 範本資料庫。

若要傳回以引號括住的 CATNAME 欄，請使用下列範例。

```
SELECT catid, QUOTE_IDENT(catname)
```

```
FROM category
ORDER BY 1,2;

+-------+-------------+
| catid | quote_ident |
+-------+-------------+
|     1 | "MLB"       |
|     2 | "NHL"       |
|     3 | "NFL"       |
|     4 | "NBA"       |
|     5 | "MLS"       |
|     6 | "Musicals"  |
|     7 | "Plays"     |
|     8 | "Opera"     |
|     9 | "Pop"       |
|    10 | "Jazz"      |
|    11 | "Classical" |
+-------+-------------+
```

QUOTE_LITERAL 函數

QUOTE_LITERAL 函數將指定的字串做為單引號括住的字串傳回，可供 SQL 陳述式中做為字串常值。如果輸入參數是數字，QUOTE_LITERAL 會將輸入參數視為字串。適當地將任何內嵌單引號和反斜線加倍。

語法

```
QUOTE_LITERAL(string)
```

引數

string

CHAR 或 VARCHAR 字串。

傳回類型

QUOTE_LITERAL 函數傳回與輸入字串的資料類型相同的 CHAR 或 VARCHAR 字串。

範例

若要傳回含有單引號的字串 ''CAT''，請使用下列範例。

```
SELECT QUOTE_LITERAL('''CAT''');

+---------------+
| quote_literal |
+---------------+
| '''CAT'''     |
+---------------+
```

下列範例會使用 TICKIT 範例資料庫中 CATEGORY 表格中的資料。如需詳細資訊，請參閱 範本資料庫。

若要傳回以單引號括住的 CATNAME 欄，請使用下列範例。

```
SELECT catid, QUOTE_LITERAL(catname)
FROM category
ORDER BY 1,2;

+-------+---------------+
| catid | quote_literal |
+-------+---------------+
|     1 | 'MLB'         |
|     2 | 'NHL'         |
|     3 | 'NFL'         |
|     4 | 'NBA'         |
|     5 | 'MLS'         |
|     6 | 'Musicals'    |
|     7 | 'Plays'       |
|     8 | 'Opera'       |
|     9 | 'Pop'         |
|    10 | 'Jazz'        |
|    11 | 'Classical'   |
+-------+---------------+
```

若要傳回以單引號括住的 CATID 欄，請使用下列範例。

```
SELECT QUOTE_LITERAL(catid), catname
FROM category
ORDER BY 1,2;

+---------------+-----------+
| quote_literal |  catname  |
+---------------+-----------+
```

```
| '1'           | MLB       |
| '10'          | Jazz      |
| '11'          | Classical |
| '2'           | NHL       |
| '3'           | NFL       |
| '4'           | NBA       |
| '5'           | MLS       |
| '6'           | Musicals  |
| '7'           | Plays     |
| '8'           | Opera     |
| '9'           | Pop       |
+---------------+-----------+
```

REGEXP_COUNT 函數

在字串中搜尋規則運算式模式，並傳回整數指出所指定模式出現在字串中的次數。如果找不到相符項目，則函數會傳回 0。有關正則表達式的更多信息，請參閱POSIX 運算子維基百科中的正則表達式。

語法

```
REGEXP_COUNT( source_string, pattern [, position [, parameters ] ] )
```

引數

source_string

CHAR 或 VARCHAR 字串。

pattern

代表規則運算式模式的 UTF-8 字串常值。如需詳細資訊，請參閱 POSIX 運算子。

position

(選用) 正 INTEGER，表示在 source_string 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設值為 1。如果 position 小於 1，則從 source_string 的第一個字元開始搜尋。如果 position 大於 source_string 中的字元數，則結果為 0。

參數

(選用) 一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。

- i - 進行不區分大小寫比對。
- p - 使用 Perl 相容規則運算式 (PCRE) 方言解釋此模式。有關 PCRE 的更多信息，請參閱維基百科中的 Perl 兼容正則表達式。

傳回類型

INTEGER

範例

若要計算三字母序列出現的次數，請使用下列範例。

```
SELECT REGEXP_COUNT('abcdefghijklmnopqrstuvwxyz', '[a-z]{3}');

+--------------+
| regexp_count |
+--------------+
|            8 |
+--------------+
```

若要使用不區分大小寫的比對來計算字串 FOX 的出現次數，請使用下列範例。

```
SELECT REGEXP_COUNT('the fox', 'FOX', 1, 'i');

+--------------+
| regexp_count |
+--------------+
|            1 |
+--------------+
```

若要使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞，請使用下列範例。此範例使用 ?= 運算子，該運算子在 PCRE 中具有特定的前瞻內涵。此範例會計算此類字詞的出現次數，並使用區分大小寫的比對。

```
SELECT REGEXP_COUNT('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 'p');

+--------------+
| regexp_count |
+--------------+
|            2 |
```

```
+--------------+
```

若要使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞，請使用下列範例。它使用 ?= 運算子，該運算子在 PCRE 中具有特定的內涵。此範例會計算此類字詞的出現次數，但與前一個範例不同，因為它使用不區分大小寫的比對。

```
SELECT REGEXP_COUNT('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 'ip');

+--------------+
| regexp_count |
+--------------+
|            3 |
+--------------+
```

下列範例會使用 TICKIT 範例資料庫中 USERS 資料表的資料。如需詳細資訊，請參閱 範本資料庫。

若要計算頂層網域名稱為 org 或 edu 的次數，請使用下列範例。

```
SELECT email, REGEXP_COUNT(email,'@[^.]*\.(org|edu)') FROM users
ORDER BY userid LIMIT 4;

+----------------------------------------------+--------------+
|                    email                     | regexp_count |
+----------------------------------------------+--------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu |            1 |
| Suspendisse.tristique@nonnisiAenean.edu       |            1 |
| amet.faucibus.ut@condimentumegetvolutpat.ca   |            0 |
| sed@lacusUtnec.ca                             |            0 |
+----------------------------------------------+--------------+
```

REGEXP_INSTR 函數

在字串中搜尋規則表達式模式，並傳回整數指出相符子字串的開始位置或結尾位置。如果找不到相符項目，則函數會傳回 0。REGEXP_INSTR 類似於 POSITION 函數，但可讓您在字串中搜尋規則表達式模式。有關正則表達式的更多信息，請參閱POSIX 運算子維基百科中的正則表達式。

語法

```
REGEXP_INSTR( source_string, pattern [, position [, occurrence] [, option [, parameters
 ] ] ] ] )
```

引數

source_string

要搜尋的字串表達式，例如欄名。

pattern

代表規則運算式模式的 UTF-8 字串常值。如需詳細資訊，請參閱 POSIX 運算子。

position

(選用) 正 INTEGER，表示在 source_string 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設值為 1。如果 position 小於 1，則從 source_string 的第一個字元開始搜尋。如果 position 大於 source_string 中的字元數，則結果為 0。

occurrence

(選用) 正 INTEGER，表示要使用哪一個出現的模式。REGEXP_INSTR 略過前 *occurrence*-1 個相符項目。預設值為 1。如果 occurrence 小於 1 或大於 source_string 中的字元數，則忽略搜尋，且結果為 0。

option

(選用) 此值指出要傳回相符項目第一個字元的位置 (0)，還是相符項目後第一個字元的位置 (1)。非零值與 1 相同。預設值為 0。

參數

(選用) 一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。
- i - 進行不區分大小寫比對。
- e - 使用子運算式擷取子字串。

  如果 pattern 包含子表達式，REGEXP_INSTR 使用 pattern 中的第一個子表達式來比對子字串。REGEXP_INSTR 只考慮第一個子表達式；忽略其他子表達式。如果模式沒有子表達式，REGEXP_INSTR 會忽略 'e' 參數。
- p - 使用 Perl 相容規則運算式 (PCRE) 方言解釋此模式。有關 PCRE 的更多信息，請參閱維基百科中的 Perl 兼容正則表達式。

傳回類型

Integer

範例

下列範例使用 TICKIT 範例資料庫中 USERS 資料表的資料。如需詳細資訊，請參閱 範本資料庫。

若要搜尋網域名稱開頭的 @ 字元，並傳回第一個相符項目的開始位置，請使用以下範例。

```
SELECT email, REGEXP_INSTR(email, '@[^.]*')
FROM users
ORDER BY userid LIMIT 4;

+----------------------------------------------+--------------+
|                     email                    | regexp_instr |
+----------------------------------------------+--------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu |           21 |
| Suspendisse.tristique@nonnisiAenean.edu       |           22 |
| amet.faucibus.ut@condimentumegetvolutpat.ca   |           17 |
| sed@lacusUtnec.ca                             |            4 |
+----------------------------------------------+--------------+
```

若要搜尋單字 Center 的變體，並傳回第一個相符項目的開始位置，請使用以下範例。

```
SELECT venuename, REGEXP_INSTR(venuename,'[cC]ent(er|re)$')
FROM venue
WHERE REGEXP_INSTR(venuename,'[cC]ent(er|re)$') > 0
ORDER BY venueid LIMIT 4;

+-----------------------+--------------+
|       venuename       | regexp_instr |
+-----------------------+--------------+
| The Home Depot Center |           16 |
| Izod Center           |            6 |
| Wachovia Center       |           10 |
| Air Canada Centre     |           12 |
+-----------------------+--------------+
```

若要使用不區分大小寫的比對邏輯來尋找字串 FOX 第一次出現的開始位置，請使用下列範例。

```
SELECT REGEXP_INSTR('the fox', 'FOX', 1, 1, 0, 'i');

+--------------+
| regexp_instr |
+--------------+
```

```
|            5 |
+--------------+
```

若要使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞，請使用下列範例。它使用 ?= 運算子，該運算子在 PCRE 中具有特定的前瞻內涵。此範例會尋找第二個此類字詞的開始位置。

```
SELECT REGEXP_INSTR('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 0, 'p');

+--------------+
| regexp_instr |
+--------------+
|           21 |
+--------------+
```

若要使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞，請使用下列範例。它使用 ?= 運算子，該運算子在 PCRE 中具有特定的前瞻內涵。此範例會尋找第二個這類字詞的開始位置，但與前一個範例不同，因為它使用不區分大小寫的比對。

```
SELECT REGEXP_INSTR('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 0, 'ip');

+--------------+
| regexp_instr |
+--------------+
|           15 |
+--------------+
```

REGEXP_REPLACE 函數

在字串中搜尋規則表達式模式，並以指定的字串取代每一個出現的模式。REGEXP_REPLACE 類似於 REPLACE 函數，但可讓您在字串中搜尋規則表達式模式。有關正則表達式的更多信息，請參閱POSIX 運算子維基百科中的正則表達式。

REGEXP_REPLACE 類似於 TRANSLATE 函數 和 REPLACE 函數，但 TRANSLATE 會進行多次單一字元替換，REPLACE 會將一整個字串替換成另一個字串，而 REGEXP_REPLACE 可讓您在字串中搜尋規則表達式模式。

語法

```
REGEXP_REPLACE( source_string, pattern [, replace_string [ , position [, parameters
 ] ] ] )
```

引數

source_string

要搜尋的 CHAR 或 VARCHAR 字串運算式，例如欄名。

pattern

代表規則運算式模式的 UTF-8 字串常值。如需詳細資訊，請參閱 POSIX 運算子。

replace_string

(選用) CHAR 或 VARCHAR 字串運算式 (例如欄名)，用於搜尋每一個出現的模式。預設為空字串 ("")。

position

(選用) 正整數，表示在 source_string 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設值為 1。如果 position 小於 1，則從 source_string 的第一個字元開始搜尋。如果 position 大於 source_string 中的字元數，則結果為 source_string。

參數

(選用) 一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。
- i - 進行不區分大小寫比對。
- p - 使用 Perl 相容規則運算式 (PCRE) 方言解釋此模式。有關 PCRE 的更多信息，請參閱維基百科中的 Perl 兼容正則表達式。

傳回類型

VARCHAR

如果 pattern 或 replace_string 為 NULL，函數會傳回 NULL。

範例

若要使用不區分大小寫的比對取代值 `quick brown fox` 內所有出現的字串 FOX，請使用下列範例。

```
SELECT REGEXP_REPLACE('the fox', 'FOX', 'quick brown fox', 1, 'i');

+---------------------+
|   regexp_replace    |
+---------------------+
| the quick brown fox |
+---------------------+
```

下列範例會使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞。它使用 ?= 運算子，該運算子在 PCRE 中具有特定的前瞻內涵。若要以值 [hidden] 取代這類字詞的每個出現值，請使用下列範例。

```
SELECT REGEXP_REPLACE('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 '[hidden]', 1, 'p');

+-------------------------------+
|        regexp_replace         |
+-------------------------------+
| [hidden] plain A1234 [hidden] |
+-------------------------------+
```

下列範例會使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞。它使用 ?= 運算子，該運算子在 PCRE 中具有特定的前瞻內涵。若要以值 [hidden] 取代這類字詞的每個出現值，但與前面的範例不同之處在於它使用不區分大小寫的比對，請使用下列範例。

```
SELECT REGEXP_REPLACE('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 '[hidden]', 1, 'ip');

+---------------------------------+
|          regexp_replace         |
+---------------------------------+
| [hidden] plain [hidden] [hidden] |
+---------------------------------+
```

下列範例使用 TICKIT 範例資料庫中 USERS 資料表的資料。如需詳細資訊，請參閱 範本資料庫。

若要從電子郵件地中刪除 @ 和網域名稱，請使用下列範例。

```
SELECT email, REGEXP_REPLACE(email, '@.*\\.(org|gov|com|edu|ca)$')
FROM users
ORDER BY userid LIMIT 4;
```

```
+-----------------------------------------------+----------------------+
|                     email                     |    regexp_replace    |
+-----------------------------------------------+----------------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu | Etiam.laoreet.libero |
| Suspendisse.tristique@nonnisiAenean.edu       | Suspendisse.tristique |
| amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut     |
| sed@lacusUtnec.ca                             | sed                  |
+-----------------------------------------------+----------------------+
```

若要以 internal.company.com 取代電子郵件地址的網域名稱，請使用下列範例。

```
SELECT email, REGEXP_REPLACE(email, '@.*\\.[[:alpha:]]{2,3}','@internal.company.com')
FROM users
ORDER BY userid LIMIT 4;

+-----------------------------------------------
+-------------------------------------------+
|                     email                     |               regexp_replace
      |
+-----------------------------------------------
+-------------------------------------------+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu |
 Etiam.laoreet.libero@internal.company.com  |
| Suspendisse.tristique@nonnisiAenean.edu       |
 Suspendisse.tristique@internal.company.com |
| amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut@internal.company.com
      |
| sed@lacusUtnec.ca                             | sed@internal.company.com
      |
+-----------------------------------------------
+-------------------------------------------+
```

REGEXP_SUBSTR 函數

搜尋規則運算式模式，傳回字串中的字元。REGEXP_SUBSTR 類似於 SUBSTRING 函數 函數，但可讓您在字串中搜尋規則表達式模式。如果函數不能比對規則運算式與字串中的任何字元，則傳回一個空字串。有關正則表達式的更多信息，請參閱POSIX 運算子維基百科中的正則表達式。

語法

```
REGEXP_SUBSTR( source_string, pattern [, position [, occurrence [, parameters ] ] ] )
```

引數

source_string

要搜尋的字串運算式。

pattern

代表規則運算式模式的 UTF-8 字串常值。如需詳細資訊，請參閱 POSIX 運算子。

position

正整數，表示在 source_string 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設為 1。如果 position 小於 1，則從 source_string 的第一個字元開始搜尋。如果 position 大於 source_string 中的字元數，則結果為空字串 ("")。

occurrence

正整數，表示要使用哪一個出現的模式。REGEXP_SUBSTR 略過前 occurrence -1 個相符項目。預設為 1。如果 occurrence 小於 1 或大於 source_string 中的字元數，則忽略搜尋，且結果為 NULL。

參數

一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。
- i - 進行不區分大小寫比對。
- e - 使用子運算式擷取子字串。

  如果 pattern 包含子表達式，REGEXP_SUBSTR 使用 pattern 中的第一個子表達式來比對子字串。子運算式是用括號括起來的模式中的運算式。例如，對於模式 `'This is a '`，會比對第一個運算式和字串 `'This is a (\\w+)'` 後跟一個單詞。具有 e 參數的 REGEXP_SUBSTR 不會傳回 pattern，而是僅傳回子運算式內的字串。

  REGEXP_SUBSTR 只考慮第一個子表達式；忽略其他子表達式。如果模式沒有子表達式，REGEXP_SUBSTR 會忽略 'e' 參數。
- p - 使用 Perl 相容規則運算式 (PCRE) 方言解釋此模式。有關 PCRE 的更多信息，請參閱維基百科中的 Perl 兼容正則表達式。

傳回類型

VARCHAR

範例

下列範例傳回電子郵件地址在 @ 和網域域名之間的部分。查詢的 users 資料來自 Amazon Redshift 範例資料。如需詳細資訊，請參閱 範本資料庫。

```
SELECT email, regexp_substr(email,'@[^.]*')
FROM users
ORDER BY userid LIMIT 4;

                     email                      |      regexp_substr
------------------------------------------------+--------------------------
 Suspendisse.tristique@nonnisiAenean.edu        | @nonnisiAenean
 amet.faucibus.ut@condimentumegetvolutpat.ca    | @condimentumegetvolutpat
 sed@lacusUtnec.ca                              | @lacusUtnec
 Cum@accumsan.com                               | @accumsan
```

下列範例會使用不區分大小寫的比對，傳回與第一次出現的字串 FOX 相對應的輸入部分。

```
SELECT regexp_substr('the fox', 'FOX', 1, 1, 'i');

 regexp_substr
---------------
 fox
```

下列範例會使用不區分大小寫的比對，傳回與第二次出現的字串 FOX 相對應的輸入部分。結果為 NULL (空)，因為沒有第二次出現。

```
SELECT regexp_substr('the fox', 'FOX', 1, 2, 'i');

 regexp_substr
---------------


```

下列範例會傳回以小寫字母開頭之輸入的第一部分。這在函數上與沒有 c 參數的相同 SELECT 陳述式完全相同。

```
SELECT regexp_substr('THE SECRET CODE IS THE LOWERCASE PART OF 1931abc0EZ.', '[a-z]+',
 1, 1, 'c');

 regexp_substr
---------------
```

```
 abc
```

下列範例會使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞。它使用 ?= 運算子，該運算子在 PCRE 中具有特定的前瞻內涵。此範例會傳回對應於第二個這類字詞的輸入部分。

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 'p');

 regexp_substr
---------------
 a1234
```

下列範例會使用 PCRE 方言撰寫的模式來尋找至少包含一個數字和一個小寫字母的字詞。它使用 ?= 運算子，該運算子在 PCRE 中具有特定的前瞻內涵。此範例會傳回與第二個此類字詞對應的輸入部分，但與前一個範例不同，因為它使用不區分大小寫的比對。

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
 1, 2, 'ip');

 regexp_substr
---------------
 A1234
```

下列範例會使用子運算式，使用不區分大小寫的比對來尋找符合模式 'this is a (\\w+)' 的第二個字串。它傳回括號內的子運算式。

```
SELECT regexp_substr(
               'This is a cat, this is a dog. This is a mouse.',
               'this is a (\\w+)', 1, 2, 'ie');

 regexp_substr
---------------
 dog
```

REPEAT 函數

將字串重複指定的次數。如果輸入參數是數值，REPEAT 會將輸入參數視為字串。

REPLICATE 函數 的同義詞。

語法

```
REPEAT(string, integer)
```

引數

string

第一個輸入參數是要重複的字串。

integer

第二個參數是 INTEGER，表示字串重複的次數。

傳回類型

VARCHAR

範例

下列範例會使用 TICKIT 範例資料庫中 CATEGORY 表格中的資料。如需詳細資訊，請參閱 範本資料庫。

若要將 CATEGORY 資料表中 CATID 欄的值重複三次，請使用下列範例。

```
SELECT catid, REPEAT(catid,3)
FROM category
ORDER BY 1,2;

+-------+--------+
| catid | repeat |
+-------+--------+
|     1 |    111 |
|     2 |    222 |
|     3 |    333 |
|     4 |    444 |
|     5 |    555 |
|     6 |    666 |
|     7 |    777 |
|     8 |    888 |
|     9 |    999 |
|    10 | 101010 |
```

```
|    11 | 111111 |
+-------+--------+
```

REPLACE 函數

以其他指定的字元取代一組字元在現有字串內出現的所有地方。

REPLACE 類似於 TRANSLATE 函數 和 REGEXP_REPLACE 函數，但 TRANSLATE 會進行多次單一字元替換，REGEXP_REPLACE 可讓您在字串中搜尋規則表達式模式，而 REPLACE 會將一整個字串替換成另一個字串。

語法

```
REPLACE(string, old_chars, new_chars)
```

引數

string

要搜尋的 CHAR 或 VARCHAR 字串

old_chars

要取代的 CHAR 或 VARCHAR 字串

new_chars

新的 CHAR 或 VARCHAR 字串，用來取代 old_string。

傳回類型

VARCHAR

如果 old_chars 或 new_chars 為 NULL，則結果為 NULL。

範例

下列範例會使用 TICKIT 範例資料庫中 CATEGORY 表格中的資料。如需詳細資訊，請參閱 範本資料庫。

若要將 CATGROUP 欄位中的字串 Shows 轉換為 Theatre，請使用下列範例。

```
SELECT catid, catgroup, REPLACE(catgroup, 'Shows', 'Theatre')
FROM category
ORDER BY 1,2,3;

+-------+----------+----------+
| catid | catgroup | replace  |
+-------+----------+----------+
|     1 | Sports   | Sports   |
|     2 | Sports   | Sports   |
|     3 | Sports   | Sports   |
|     4 | Sports   | Sports   |
|     5 | Sports   | Sports   |
|     6 | Shows    | Theatre  |
|     7 | Shows    | Theatre  |
|     8 | Shows    | Theatre  |
|     9 | Concerts | Concerts |
|    10 | Concerts | Concerts |
|    11 | Concerts | Concerts |
+-------+----------+----------+
```

REPLICATE 函數

REPEAT 函數的同義詞。

請參閱REPEAT 函數。

REVERSE 函數

REVERSE 函數操作字串並傳回相反順序的字元。例如，reverse('abcde') 傳回 edcba。此函數適用於數值和日期資料類型，以及字元資料類型；不過，在大部分情況下，字元字串有實用值。

語法

```
REVERSE( expression )
```

引數

運算式

具有字元、日期、時間戳記或數值資料類型的運算式，代表字元反轉的目標。所有運算式都會隱含轉換為 VARCHAR 字串。CHAR 字串中的結尾空格會遭到忽略。

傳回類型

VARCHAR

範例

下列範例使用來自 TICKIT 範例資料庫中 USERS 和 SALES 資料表的資料。如需詳細資訊，請參閱 範本資料庫。

若要從 USERS 資料表中選取五個不同城市名稱及其對應的反轉名稱，請使用下列範例。

```
SELECT DISTINCT city AS cityname, REVERSE(cityname)
FROM users
ORDER BY city LIMIT 5;

+----------+----------+
| cityname | reverse  |
+----------+----------+
| Aberdeen | needrebA |
| Abilene  | enelibA  |
| Ada      | adA      |
| Agat     | tagA     |
| Agawam   | mawagA   |
+----------+----------+
```

若要選取五個銷售 ID 及其對應的反轉 ID (轉換為字元字串)，請使用下列範例。

```
SELECT salesid, REVERSE(salesid)
FROM sales
ORDER BY salesid DESC LIMIT 5;

+---------+---------+
| salesid | reverse |
+---------+---------+
|  172456 |  654271 |
|  172455 |  554271 |
|  172454 |  454271 |
|  172453 |  354271 |
|  172452 |  254271 |
+---------+---------+
```

RTRIM 函數

RTRIM 函數從字串結尾修剪一組指定的字元。移除只包含修剪字元清單中字元的最長字串。當修剪字元沒有出現在輸入字串中時，就會完成修剪。

語法

```
RTRIM( string, trim_chars )
```

引數

string

要修剪的字串資料行、運算式或字串常值。

trim_chars

字串欄、運算式或字串常值，代表要從 string 結尾修剪的字元。如果未指定，則會使用空格作為修剪字元。

傳回類型

與 string 引數的資料類型相同的字串。

範例

下列範例從字串 ' abc ' 中修剪開頭和結尾空格：

```
select '     abc    ' as untrim, rtrim('     abc    ') as trim;

untrim    | trim
----------+------
   abc    |    abc
```

下列範例從字串 'xyzaxyzbxyzcxyz' 中移除結尾 'xyz' 字串。結尾的 'xyz' 已移除，但出現在字串內的部分則未移除。

```
select 'xyzaxyzbxyzcxyz' as untrim,
rtrim('xyzaxyzbxyzcxyz', 'xyz') as trim;

     untrim      |   trim
-----------------+-----------
```

```
 xyzaxyzbxyzcxyz | xyzaxyzbxyzc
```

下列範例會從符合 trim _chars 清單 'tes' 中任何字元的字串 'setuphistorycassettes' 中移除結尾部分。任何出現在輸入字串結尾的 trim_chars 清單中另一個字元前的 t、e 或 s 都會被移除。

```
SELECT rtrim('setuphistorycassettes', 'tes');

     rtrim
-----------------
 setuphistoryca
```

下列範例修剪 VENUENAME 結尾出現的 'Park' 這幾個字元：

```
select venueid, venuename, rtrim(venuename, 'Park')
from venue
order by 1, 2, 3
limit 10;

venueid |         venuename          |          rtrim
--------+----------------------------+-------------------------
      1 | Toyota Park                | Toyota
      2 | Columbus Crew Stadium      | Columbus Crew Stadium
      3 | RFK Stadium                | RFK Stadium
      4 | CommunityAmerica Ballpark  | CommunityAmerica Ballp
      5 | Gillette Stadium           | Gillette Stadium
      6 | New York Giants Stadium    | New York Giants Stadium
      7 | BMO Field                  | BMO Field
      8 | The Home Depot Center      | The Home Depot Cente
      9 | Dick's Sporting Goods Park | Dick's Sporting Goods
     10 | Pizza Hut Park             | Pizza Hut
```

請注意，當 P、a、r 或 k 其中任何字元出現在 VENUENAME 的結尾時，RTRIM 會移除這些字元。

SOUNDEX 函數

SOUNDEX 函數傳回 American Soundex 值，該值由輸入字串的第一個字母後面跟著代表您指定字串的英文發音的 3 位元發音編碼組成。例如，Smith 和 Smyth 具有相同的 Soundex 值。

語法

```
SOUNDEX(string)
```

引數

string

您可以指定要轉換為 American Soundex 代碼值的 CHAR 或 VARCHAR 字串。

傳回類型

VARCHAR(4)

使用須知

SOUNDEX 函數只會轉換英文字母小寫和大寫 ASCII 字元，包括 A-z 和 A-Z。SOUNDEX 會忽略其他字元。SOUNDEX 傳回由空格分隔的多個單詞的字串的單個 Soundex 值。

```
SELECT SOUNDEX('AWS Amazon');

+---------+
| soundex |
+---------+
| A252    |
+---------+
```

SOUNDEX 傳回一個空字串，如果輸入字串不包含任何英文字母。

```
SELECT SOUNDEX('+-*/%');

+---------+
| soundex |
+---------+
|         |
+---------+
```

範例

若要傳回 Amazon 的 Soundex 值，請使用下列範例。

```
SELECT SOUNDEX('Amazon');

+---------+
| soundex |
+---------+
```

```
| A525    |
+---------+
```

若要傳回 smith 和 smyth 的 Soundex 值，請使用下列範例。請注意，聲音值是相同的。

```
SELECT SOUNDEX('smith'), SOUNDEX('smyth');

+-------+-------+
| smith | smyth |
+-------+-------+
| S530  | S530  |
+-------+-------+
```

SPLIT_PART 函數

在指定分隔符號之處分割字串，並傳回指定位置的部分。

語法

```
SPLIT_PART(string, delimiter, position)
```

引數

string

要分割的字串資料欄、運算式或字串常值。字串可以是 CHAR 或 VARCHAR。

delimiter

分隔符號字串，表示輸入 string 的部分。

如果 delimiter 是常值，請以單引號括住。

position

要傳回之字串部分的位置 (從 1 起算)。必須是大於 0 的整數。如果 position 大於字串部分的數目，SPLIT_PART 會傳回空字串。如果在 string 中找不到 delimiter，則傳回的值包含指定部分的內容，這可能是整個 string 或空值。

傳回類型

CHAR 或 VARCHAR 字串，與字串參數相同。

範例

下列範例會使用 $ 分隔符號將字串常值分割成多個部分，並傳回第二部分。

```
select split_part('abc$def$ghi','$',2)

split_part
----------
def
```

以下範例將使用 $ 分隔符號將字串常值分割成多個部分。它傳回一個空字串，因為沒有找到部分 4。

```
select split_part('abc$def$ghi','$',4)

split_part
----------

```

以下範例將使用 # 分隔符號將字串常值分割成多個部分。它傳回整個字串，這是第一部分，因為沒有找到分隔符號。

```
select split_part('abc$def$ghi','#',1)

split_part
------------
abc$def$ghi
```

下列範例將時間戳記欄位 LISTTIME 分割成年、月、日元素。

```
select listtime, split_part(listtime,'-',1) as year,
split_part(listtime,'-',2) as month,
split_part(split_part(listtime,'-',3),' ',1) as day
from listing limit 5;

      listtime       | year | month | day
---------------------+------+-------+------
 2008-03-05 12:25:29 | 2008 | 03    | 05
 2008-09-09 08:03:36 | 2008 | 09    | 09
 2008-09-26 05:43:12 | 2008 | 09    | 26
 2008-10-04 02:00:30 | 2008 | 10    | 04
 2008-01-06 08:33:11 | 2008 | 01    | 06
```

下列範例選取 LISTTIME 時間戳記欄位，依 '-' 字元分割以取得月份 (LISTTIME 字串的第二部分)，然後計算每月的項目數：

```
select split_part(listtime,'-',2) as month, count(*)
from listing
group by split_part(listtime,'-',2)
order by 1, 2;

 month | count
-------+-------
    01 | 18543
    02 | 16620
    03 | 17594
    04 | 16822
    05 | 17618
    06 | 17158
    07 | 17626
    08 | 17881
    09 | 17378
    10 | 17756
    11 | 12912
    12 | 4589
```

STRPOS 函數

傳回子字串在指定字串內的位置。

如需相似函數，請參閱 CHARINDEX 函數 和 POSITION 函數。

語法

```
STRPOS(string, substring )
```

引數

string

第一個輸入參數是要搜尋的 CHAR 或 VARCHAR 字串。

substring

第二個參數是在 string 內要搜尋的子字串。

傳回類型

INTEGER

STRPOS 函數傳回對應於子字串位置的 INTEGER (以 1 開始，不是以零開始)。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。

使用須知

如果在 string 內找不到 substring，STRPOS 會傳回 0。

```
SELECT STRPOS('dogfish', 'fist');

+--------+
| strpos |
+--------+
|      0 |
+--------+
```

範例

若要顯示 fish 內 dogfish 的位置，請使用下列範例。

```
SELECT STRPOS('dogfish', 'fish');

+--------+
| strpos |
+--------+
|      4 |
+--------+
```

下列範例會使用 TICKIT 範例資料庫中 SALES 資料表的資料。如需詳細資訊，請參閱 範本資料庫。

若要從 SALES 資料表中傳回 COMMISSION 超過 999.00 的銷售交易次數，請使用下列範例。

```
SELECT DISTINCT STRPOS(commission, '.'),
COUNT (STRPOS(commission, '.'))
FROM sales
WHERE STRPOS(commission, '.') > 4
GROUP BY STRPOS(commission, '.')
ORDER BY 1, 2;
```

```
+--------+-------+
| strpos | count |
+--------+-------+
|      5 |   629 |
+--------+-------+
```

STRTOL 函數

將指定底數之數字的字串表達式轉換為等同的整數值。轉換的值必須在帶正負號的 64 位元範圍內。

語法

```
STRTOL(num_string, base)
```

引數

num_string

要轉換之數字的字串表達式。如果 num_string 是空的 ('')，或開頭為 Null 字元 ('\0')，則轉換的值為 0。如果 num_string 是含有 NULL 值的欄，STRTOL 會傳回 NULL。字串的開頭可以有任何數量的空格，後面可選擇接著單一加號 '+' 或減號 '-' 來表示正或負。預設為 '+'。如果 base 是 16，字串可選擇以 '0x' 開頭。

base

INTEGER 介於 2 和 36 之間。

傳回類型

BIGINT

如果 num_string 為空，則函數傳回 NULL。

範例

若要將字串和基值對轉換為整數，請使用下列範例。

```
SELECT STRTOL('0xf',16);

+--------+
| strtol |
+--------+
```

```
|     15 |
+--------+

SELECT STRTOL('abcd1234',16);

+------------+
|   strtol   |
+------------+
| 2882343476 |
+------------+

SELECT STRTOL('1234567', 10);

+---------+
| strtol  |
+---------+
| 1234567 |
+---------+

SELECT STRTOL('1234567', 8);

+--------+
| strtol |
+--------+
| 342391 |
+--------+

SELECT STRTOL('110101', 2);

+--------+
| strtol |
+--------+
|     53 |
+--------+

SELECT STRTOL('\0', 2);

+--------+
| strtol |
+--------+
|      0 |
+--------+
```

SUBSTRING 函數

根據指定的開始位置傳回字串的子集。

如果輸入是字串，則提取的開始位置和字元數是以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。如果輸入是二進位運算式，則開始位置和提取的子字串是以位元組為基礎。您不能指定負長度，但可以指定負的開始位置。

語法

```
SUBSTRING(character_string FROM start_position [ FOR number_characters ] )
```

```
SUBSTRING(character_string, start_position, number_characters )
```

```
SUBSTRING(binary_expression, start_byte, number_bytes )
```

```
SUBSTRING(binary_expression, start_byte )
```

引數

character_string

供進行搜尋的字串。非字元資料類型視為字串。

start_position

在字串內要開始擷取的位置，從 1 開始。start_position 以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。這可以是負數。

number_characters

要擷取的字元數 (子字串的長度)。number_characters 以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。這不能是負數。

binary_expression

要搜尋之資料類型 VARBYTE 的 binary_expression。

start_byte

在二進位運算式內要開始擷取的位置，從 1 開始。這可以是負數。

number_bytes

要擷取的位元組數，即子字串的長度。此數字不可以是負數。

傳回類型

VARCHAR 或 VARBYTE 取決於輸入。

使用須知

以下是如何使用 start_position 和 number_characters 從字串中的各個位置提取子字串的一些範例。

下列範例傳回從第六個字元開始的四個字元的字串。

```
select substring('caterpillar',6,4);
substring
-----------
pill
(1 row)
```

如果 start_position + number_characters 超過 string 的長度，SUBSTRING 會傳回從 start_position 開始到字串結尾的字串。例如：

```
select substring('caterpillar',6,8);
substring
-----------
pillar
(1 row)
```

如果 start_position 是負數或 0，SUBSTRING 函數會傳回從字串第一個字元開始且長度為 start_position + number_characters -1 的子字串。例如：

```
select substring('caterpillar',-2,6);
substring
-----------
cat
(1 row)
```

如果 start_position + number_characters -1 小於或等於零，SUBSTRING 會傳回空字串。例如：

```
select substring('caterpillar',-5,4);
substring
-----------
```

```
(1 row)
```

範例

下列範例從 LISTING 資料表的 LISTTIME 字串中傳回月份：

```
select listid, listtime,
substring(listtime, 6, 2) as month
from listing
order by 1, 2, 3
limit 10;

 listid |      listtime       | month
--------+---------------------+-------
      1 | 2008-01-24 06:43:29 | 01
      2 | 2008-03-05 12:25:29 | 03
      3 | 2008-11-01 07:35:33 | 11
      4 | 2008-05-24 01:18:37 | 05
      5 | 2008-05-17 02:29:11 | 05
      6 | 2008-08-15 02:08:13 | 08
      7 | 2008-11-15 09:38:15 | 11
      8 | 2008-11-09 05:07:30 | 11
      9 | 2008-09-09 08:03:36 | 09
     10 | 2008-06-17 09:44:54 | 06
(10 rows)
```

下列範例同上，但使用 FROM...FOR 選項：

```
select listid, listtime,
substring(listtime from 6 for 2) as month
from listing
order by 1, 2, 3
limit 10;

 listid |      listtime       | month
--------+---------------------+-------
      1 | 2008-01-24 06:43:29 | 01
      2 | 2008-03-05 12:25:29 | 03
      3 | 2008-11-01 07:35:33 | 11
      4 | 2008-05-24 01:18:37 | 05
      5 | 2008-05-17 02:29:11 | 05
      6 | 2008-08-15 02:08:13 | 08
      7 | 2008-11-15 09:38:15 | 11
```

```
      8 | 2008-11-09 05:07:30 | 11
      9 | 2008-09-09 08:03:36 | 09
     10 | 2008-06-17 09:44:54 | 06
(10 rows)
```

如果字串可能包含雙位元組字元，則您無法使用 SUBSTRING 來肯定地擷取字串的字首，因為您需要根據位元組數來指定雙位元組字串的長度，而不是字元數。若要根據位元組長度來擷取字串的開頭部分，您可以將字串 CAST 成為 VARCHAR(byte_length) 以截斷字串，其中 byte_length 是所需的長度。下列範例從 'Fourscore and seven' 字串中擷取前 5 個位元組。

```
select cast('Fourscore and seven' as varchar(5));

varchar
-------
Fours
```

下列範例顯示二進位值 abc 的負開始位置。因為開始位置是 -3，所以子字串是從二進位值的開頭提取的。結果自動顯示為二進位子字串的十六進位表示形式。

```
select substring('abc'::varbyte, -3);

 substring
-----------
 616263
```

下列範例顯示 1 作為二進位值 abc 的開始位置。因為沒有指定長度，所以字串從開始位置提取到字串的結尾。結果自動顯示為二進位子字串的十六進位表示形式。

```
select substring('abc'::varbyte, 1);

 substring
-----------
 616263
```

下列範例顯示 3 作為二進位值 abc 的開始位置。因為沒有指定長度，所以字串是從開始位置提取到字串的結尾。結果自動顯示為二進位子字串的十六進位表示形式。

```
select substring('abc'::varbyte, 3);

 substring
```

```
-----------
 63
```

下列範例顯示 2 作為二進位值 abc 的開始位置。字串從開始位置提取到位置 10，但字串的結尾位於位置 3。結果自動顯示為二進位子字串的十六進位表示形式。

```
select substring('abc'::varbyte, 2, 10);

 substring
-----------
 6263
```

下列範例顯示 2 作為二進位值 abc 的開始位置。字串從開始位置提取 1 個位元組。結果自動顯示為二進位子字串的十六進位表示形式。

```
select substring('abc'::varbyte, 2, 1);

 substring
-----------
 62
```

下列範例會傳回輸入字串 Silva, Ana 中最後一個空格之後出現的名字 Ana。

```
select reverse(substring(reverse('Silva, Ana'), 1, position(' ' IN reverse('Silva,
 Ana'))))

 reverse
-----------
 Ana
```

TEXTLEN 函數

LEN 函數的同義詞。

請參閱LEN 函數。

TRANSLATE 函數

對於給定的表達式，以指定的替換值取代所有出現的指定字元。現有字元依位置映射至 characters_to_replace 和 characters_to_substitute 引數中的替換字元。如果 characters_to_replace 引

數中指定的字元數比 characters_to_substitute 引數更多，傳回值中會省略 characters_to_replace 引數中額外的字元。

TRANSLATE 類似於 REPLACE 函數 和 REGEXP_REPLACE 函數，但 REPLACE 會將一整個字串替換成另一個字串，REGEXP_REPLACE 可讓您在字串中搜尋規則表達式模式，而 TRANSLATE 會進行多次單一字元替換。

如果任何引數為 Null，則傳回值為 NULL。

語法

```
TRANSLATE( expression, characters_to_replace, characters_to_substitute )
```

引數

運算式

要轉換的表達式。

characters_to_replace

包含要取代之字元的字串。

characters_to_substitute

包含替換字元的字串。

傳回類型

VARCHAR

範例

若要取代字串中的幾個字元，請使用下列範例。

```
SELECT TRANSLATE('mint tea', 'inea', 'osin');

+-----------+
| translate |
+-----------+
| most tin  |
+-----------+
```

下列範例使用 TICKIT 範例資料庫中 USERS 資料表的資料。如需詳細資訊，請參閱 範本資料庫。

若要對某欄的所有值，以點取代 at 符號 (@)，請使用下列範例。

```
SELECT email, TRANSLATE(email, '@', '.') as obfuscated_email
FROM users LIMIT 10;

+----------------------------------------+----------------------------------------+
|                 email                  |           obfuscated_email             |
+----------------------------------------+----------------------------------------+
| Cum@accumsan.com                       | Cum.accumsan.com                       |
| lorem.ipsum@Vestibulumante.com         | lorem.ipsum.Vestibulumante.com         |
| non.justo.Proin@ametconsectetuer.edu   | non.justo.Proin.ametconsectetuer.edu   |
| non.ante.bibendum@porttitortellus.org  | non.ante.bibendum.porttitortellus.org  |
| eros@blanditatnisi.org                 | eros.blanditatnisi.org                 |
| augue@Donec.ca                         | augue.Donec.ca                         |
| cursus@pedeacurna.edu                  | cursus.pedeacurna.edu                  |
| at@Duis.com                            | at.Duis.com                            |
| quam@facilisisvitaeorci.ca             | quam.facilisisvitaeorci.ca             |
| mi.lorem@nunc.edu                      | mi.lorem.nunc.edu                      |
+----------------------------------------+----------------------------------------+
```

若要對某欄的所有值，以點底線取代空格並剔除點，請使用下列範例。

```
SELECT city, TRANSLATE(city, ' .', '_')
FROM users
WHERE city LIKE 'Sain%' OR city LIKE 'St%'
GROUP BY city
ORDER BY city;

+----------------+---------------+
|      city      |   translate   |
+----------------+---------------+
| Saint Albans   | Saint_Albans  |
| Saint Cloud    | Saint_Cloud   |
| Saint Joseph   | Saint_Joseph  |
| Saint Louis    | Saint_Louis   |
| Saint Paul     | Saint_Paul    |
| St. George     | St_George     |
| St. Marys      | St_Marys      |
| St. Petersburg | St_Petersburg |
| Stafford       | Stafford      |
| Stamford       | Stamford      |
| Stanton        | Stanton       |
| Starkville     | Starkville    |
```

```
| Statesboro     | Statesboro    |
| Staunton       | Staunton      |
| Steubenville   | Steubenville  |
| Stevens Point  | Stevens_Point |
| Stillwater     | Stillwater    |
| Stockton       | Stockton      |
| Sturgis        | Sturgis       |
+----------------+---------------+
```

TRIM 函數

以空白或指定的字元來修剪字串。

語法

```
TRIM( [ BOTH | LEADING | TRAILING ] [trim_chars FROM ] string )
```

引數

BOTH | LEADING | TRAILING

(選用) 指定從何處修剪字元。使用 BOTH 可移除開頭字元和結尾字元，使用 LEADING 僅移除開頭字元，使用 TRAILING 僅移除結尾字元。如果省略此參數，會同時修剪開頭和結尾字元。

trim_chars

(選用) 要從字串中修剪的字元。如果省略此參數，則會修剪空格。

string

要修剪的字串。

傳回類型

TRIM 函數傳回 VARCHAR 或 CHAR 字串。如果您搭配 SQL 命令來使用 TRIM 函數，Amazon Redshift 會隱含地將結果轉換為 VARCHAR。如果您在 SQL 函數的 SELECT 清單中使用 TRIM 函數，Amazon Redshift 不會隱含地轉換結果，您可能需要執行明確轉換，以避免資料類型不符的錯誤。如需明確轉換的相關資訊，請參閱 CAST 函數及 CONVERT 函數。

範例

若要從字串　dog　中修剪開頭和結尾空格，請使用下列範例。

```
SELECT TRIM('    dog    ');

+-------+
| btrim |
+-------+
| dog   |
+-------+
```

若要從字串 dog 中修剪開頭和結尾空格，請使用下列範例。

```
SELECT TRIM(BOTH FROM '    dog    ');

+-------+
| btrim |
+-------+
| dog   |
+-------+
```

若要從字串 "dog" 中移除開頭雙引號，請使用下列範例。

```
SELECT TRIM(LEADING '"' FROM'"dog"');

+-------+
| ltrim |
+-------+
| dog"  |
+-------+
```

若要從字串 "dog" 中移除結尾的雙引號，請使用下列範例。

```
SELECT TRIM(TRAILING '"' FROM'"dog"');

+-------+
| rtrim |
+-------+
| "dog  |
+-------+
```

當 trim_chars 中任何字元出現在 string 開頭或結尾時，TRIM 會移除這些字元。下列範例修剪 VENUENAME (這是 VARCHAR 欄) 開頭或結尾出現的 'C'、'D' 和 'G' 字元。如需詳細資訊，請參閱 VENUE 資料表。

```
SELECT venueid, venuename, TRIM('CDG' FROM venuename)
FROM venue
WHERE venuename LIKE '%Park'
ORDER BY 2
LIMIT 7;

+---------+----------------------------+----------------------------+
| venueid |         venuename          |           btrim            |
+---------+----------------------------+----------------------------+
|     121 | AT&T Park                  | AT&T Park                  |
|     109 | Citizens Bank Park         | itizens Bank Park          |
|     102 | Comerica Park              | omerica Park               |
|       9 | Dick's Sporting Goods Park | ick's Sporting Goods Park  |
|      97 | Fenway Park                | Fenway Park                |
|     112 | Great American Ball Park   | reat American Ball Park    |
|     114 | Miller Park                | Miller Park                |
+---------+----------------------------+----------------------------+
```

UPPER 函數

將字串轉換成大寫。UPPER 支援 UTF-8 多位元組字元，每個字元最多 4 個位元組。

語法

```
UPPER(string)
```

引數

string

輸入參數是 VARCHAR 字串或任何其他資料類型，例如 CHAR，可以隱含轉換為 VARCHAR。

傳回類型

UPPER 函數傳回與輸入字串的資料類型相同的字元字串。例如，如果輸入是 VARCHAR 字串，該函數將傳回 VARCHAR 字串。

範例

下列範例會使用 TICKIT 範例資料庫中 CATEGORY 表格中的資料。如需詳細資訊，請參閱 範本資料庫。

若要將 CATNAME 欄位轉換為大寫，請使用下列範例。

```
SELECT catname, UPPER(catname)
FROM category
ORDER BY 1,2;

+-----------+-----------+
|  catname  |   upper   |
+-----------+-----------+
| Classical | CLASSICAL |
| Jazz      | JAZZ      |
| MLB       | MLB       |
| MLS       | MLS       |
| Musicals  | MUSICALS  |
| NBA       | NBA       |
| NFL       | NFL       |
| NHL       | NHL       |
| Opera     | OPERA     |
| Plays     | PLAYS     |
| Pop       | POP       |
+-----------+-----------+
```

SUPER 類型資訊函數

接下來，您可以找到 Amazon Redshift 支援的 SQL 類型資訊函數的說明，以便從 SUPER 資料類型的輸入衍生動態資訊。

主題

- DECIMAL_PRECISION 函數
- DECIMAL_SCALE 函數
- IS_ARRAY 函數
- IS_BIGINT 函數
- IS_BOOLEAN 函數
- IS_CHAR 函數
- IS_DECIMAL 函數
- IS_FLOAT 函數
- IS_INTEGER 函數
- IS_OBJECT 函數

- IS_SCALAR 函數
- IS_SMALLINT 函數
- IS_VARCHAR 函數
- JSON_SIZE 函數
- JSON_TYPEOF 函數
- SIZE

DECIMAL_PRECISION 函數

檢查要儲存的最大小數位數的精確度。這個數字包括小數點左邊和右邊的數字。精確度的範圍是從 1 到 38，預設值為 38。

語法

```
DECIMAL_PRECISION(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

INTEGER

範例

若要將 DECIMAL_PRECISION 函數套用至資料表 t，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_PRECISION(s) FROM t;

+-------------------+
| decimal_precision |
+-------------------+
```

```
|                 6 |
+-------------------+
```

DECIMAL_SCALE 函數

檢查要儲存到小數點右邊的小數位數。刻度的範圍是從 0 到精確度點，預設值為 0。

語法

```
DECIMAL_SCALE(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

INTEGER

範例

若要將 DECIMAL_SCALE 函數套用至資料表 t，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_SCALE(s) FROM t;

+---------------+
| decimal_scale |
+---------------+
|             5 |
+---------------+
```

IS_ARRAY 函數

檢查變數是否為陣列。如果變數是陣列，函數會傳回 `true`。該函數還包括空陣列。否則，函數會針對所有其他值傳回 `false`，包括 null。

語法

```
IS_ARRAY(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_ARRAY 函數檢查 [1,2] 是否為陣列，請使用下列範例。

```
SELECT IS_ARRAY(JSON_PARSE('[1,2]'));

+----------+
| is_array |
+----------+
| true     |
+----------+
```

IS_BIGINT 函數

檢查值是否為 BIGINT。IS_BIGINT 函數會針對 64 位元範圍內標度 0 的數字傳回 true。否則，函數會針對所有其他值傳回 false，包括 null 和浮點數。

IS_BIGINT 函數是 IS_INTEGER 的超集。

語法

```
IS_BIGINT(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_BIGINT 函數檢查 5 是否為 BIGINT，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_BIGINT(s) FROM t;

+---+-----------+
| s | is_bigint |
+---+-----------+
| 5 | true      |
+---+-----------+
```

IS_BOOLEAN 函數

檢查值是否為 BOOLEAN。IS_BOOLEAN 函數會針對常數 JSON 布林值傳回 true。函數會針對所有其他值傳回 false，包括 null。

語法

```
IS_BOOLEAN(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_BOOLEAN 函數檢查 TRUE 是否為 BOOLEAN，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (TRUE);

SELECT s, IS_BOOLEAN(s) FROM t;

+------+------------+
|  s   | is_boolean |
+------+------------+
| true | true       |
+------+------------+
```

IS_CHAR 函數

檢查值是否為 CHAR。對於僅包含 ASCII 字元的字串，IS_CHAR 函數傳回 `true`，因為 CHAR 類型只能儲存 ASCII 格式的字元。函數會針對所有其他值傳回 `false`。

語法

```
IS_CHAR(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_CHAR 函數檢查 `t` 是否為 CHAR，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('t');

SELECT s, IS_CHAR(s) FROM t;
```

```
+-----+---------+
|  s  | is_char |
+-----+---------+
| "t" | true    |
+-----+---------+
```

IS_DECIMAL 函數

檢查值是否為 DECIMAL。IS_DECIMAL 函數會針對非浮點數的數字傳回 `true`。函數會針對所有其他值傳回 `false`，包括 null。

IS_DECIMAL 函數是 IS_BIGINT 的超集。

語法

```
IS_DECIMAL(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_DECIMAL 函數檢查 1.22 是否為 DECIMAL，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (1.22);

SELECT s, IS_DECIMAL(s) FROM t;

+------+------------+
|  s   | is_decimal |
+------+------------+
| 1.22 | true       |
+------+------------+
```

IS_FLOAT 函數

檢查一個值是否為浮點數。IS_FLOAT 函數會針對浮點數 (FLOAT4 和 FLOAT8) 傳回 `true`。函數會針對所有其他值傳回 `false`。

IS_DECIMAL 的集合和 IS_FLOAT 的集合是不相交的。

語法

```
IS_FLOAT(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_FLOAT 函數檢查 `2.22::FLOAT` 是否為 FLOAT，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES(2.22::FLOAT);

SELECT s, IS_FLOAT(s) FROM t;

+---------+----------+
|    s    | is_float |
+---------+----------+
| 2.22e+0 | true     |
+---------+----------+
```

IS_INTEGER 函數

對於 32 位元範圍內的小數位數為 0 的數字，則傳回 `true`；對於其他任何數字 (包括 null 和浮點數)，則傳回 `false`。

IS_INTEGER 函數是 IS_SMALLINT 函數的超集。

語法

```
IS_INTEGER(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_INTEGER 函數檢查 5 是否為 INTEGER，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_INTEGER(s) FROM t;

+---+------------+
| s | is_integer |
+---+------------+
| 5 | true       |
+---+------------+
```

IS_OBJECT 函數

檢查變數是否為物件。IS_OBJECT 函數會針對物件傳回 `true`，包括空物件。函數會針對所有其他值傳回 `false`，包括 null。

語法

```
IS_OBJECT(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_OBJECT 函數檢查 {"name": "Joe"} 是否為物件，請使用下列範例。

```
CREATE TABLE t(s super);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));

SELECT s, IS_OBJECT(s) FROM t;

+----------------+-----------+
|       s        | is_object |
+----------------+-----------+
| {"name":"Joe"} | true      |
+----------------+-----------+
```

IS_SCALAR 函數

檢查變數是否為純量。IS_SCALAR 函數會為任何非陣列或物件的值傳回 `true`。函數會針對所有其他值傳回 `false`，包括 null。

IS_ARRAY、IS_OBJECT 和 IS_SCALAR 的集合涵蓋了除 null 值之外的所有值。

語法

```
IS_SCALAR(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要檢查 {"name": "Joe"} 是否為使用 IS_SCALAR 函數的純量，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));

SELECT s, IS_SCALAR(s.name) FROM t;

+----------------+-----------+
|       s        | is_scalar |
+----------------+-----------+
| {"name":"Joe"} | true      |
+----------------+-----------+
```

IS_SMALLINT 函數

檢查變數是否為 SMALLINT。IS_SMALLINT 函數會針對 16 位元範圍內標度 0 的數字傳回 true。函數會針對所有其他值傳回 false，包括 null 和浮點數。

語法

```
IS_SMALLINT(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回

BOOLEAN

範例

若要使用 IS_SMALLINT 函數檢查 5 是否為 SMALLINT，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_SMALLINT(s) FROM t;

+---+-------------+
| s | is_smallint |
+---+-------------+
| 5 | true        |
+---+-------------+
```

IS_VARCHAR 函數

檢查變數是否為 VARCHAR。IS_VARCHAR 函數會針對所有字串傳回 `true`。函數會針對所有其他值傳回 `false`。

IS_VARCHAR 函數是 IS_CHAR 函數的超集。

語法

```
IS_VARCHAR(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_VARCHAR 函數檢查 abc 是否為 VARCHAR，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('abc');

SELECT s, IS_VARCHAR(s) FROM t;
```

```
+-------+------------+
|   s   | is_varchar |
+-------+------------+
| "abc" | true       |
+-------+------------+
```

JSON_SIZE 函數

JSON_SIZE 函數傳回傳回序列化為字串時給定 SUPER 運算式中的位元組數。

語法

```
JSON_SIZE(super_expression)
```

引數

super_expression

SUPER 常數或運算式。

傳回類型

INTEGER

JSON_SIZE 函數傳回 INTEGER，表示輸入字串中的位元組數。此值與字元數目不同。例如，UTF-8 字元 # (黑點) 的大小為 3 個位元組，即使它是 1 個字元。

使用須知

JSON_SIZE(x) 在功能上與 OCTET_LENGTH(JSON_SERIALIZE) 相同。但是，請注意，當提供的 SUPER 運算式在序列化時超過系統 VARCHAR 限制時，JSON_SERIALIZE 會傳回錯誤。JSON_SIZE 則沒有此限制。

範例

若要傳回序列化為字串的 SUPER 值的長度，請使用下列範例。

```
SELECT JSON_SIZE(JSON_PARSE('[10001,10002,"#"]'));

+-----------+
```

```
| json_size |
+-----------+
|        19 |
+-----------+
```

請注意，提供的 SUPER 運算式長度為 17 個字元，但 # 是 3 位元組字元，因此 JSON_SIZE 傳回 19。

JSON_TYPEOF 函數

JSON_TYPEOF 純量函數會傳回具有布林值、數字、字串、物件、陣列或 null 的 VARCHAR，取決於 SUPER 值的動態類型。

語法

```
JSON_TYPEOF(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

VARCHAR

範例

若要使用 JSON_TYPEOF 函數檢查陣列 [1,2] 的 JSON 類型，請使用下列範例。

```
SELECT JSON_TYPEOF(ARRAY(1,2));

+-------------+
| json_typeof |
+-------------+
| array       |
+-------------+
```

若要使用 JSON_TYPEOF 函數檢查物件 {"name":"Joe"} 的 JSON 類型，請使用下列範例。

```
SELECT JSON_TYPEOF(JSON_PARSE('{"name":"Joe"}'));
```

```
+-------------+
| json_typeof |
+-------------+
| object      |
+-------------+
```

SIZE

以 INTEGER 形式傳回 SUPER 類型常數或運算式的二進位記憶體內大小。

語法

```
SIZE(super_expression)
```

引數

super_expression

SUPER 類型常數或運算式。

傳回類型

INTEGER

範例

若要使用 SIZE 取得數個 SUPER 類型運算式的記憶體內大小，請使用下列範例。

```
CREATE TABLE test_super_size(a SUPER);

INSERT INTO test_super_size
VALUES
  (null),
  (TRUE),
  (JSON_PARSE('[0,1,2,3]')),
  (JSON_PARSE('{"a":0,"b":1,"c":2,"d":3}'))
;

SELECT a, SIZE(a)
FROM test_super_size
ORDER BY 2, 1;
```

```
+---------------------------+------+
|             a             | size |
+---------------------------+------+
| true                      |    4 |
| NULL                      |    4 |
| [0,1,2,3]                 |   23 |
| {"a":0,"b":1,"c":2,"d":3} |   52 |
+---------------------------+------+
```

VARBYTE 函數與運算子

支援 VARBYTE 資料類型的 Amazon Redshift 函數和運算子包括：

- VARBYTE 運算子
- FROM_HEX
- FROM_VARBYTE
- GETBIT
- TO_HEX
- TO_VARBYTE
- CONCAT
- LEN
- LENGTH 函數
- OCTET_LENGTH
- SUBSTRING 函數

VARBYTE 運算子

下表會列出 VARBYTE 運算子。此運算子使用資料類型 VARBYTE 的二進位值。如果一個或兩個輸入都為 null，則結果為 null。

支援的運算子

運算子	描述	傳回類型
<	小於	BOOLEAN
<=	小於或等於	BOOLEAN

運算子	描述	傳回類型
=	等於	BOOLEAN
>	大於	BOOLEAN
>=	大於或等於	BOOLEAN
!= 或 <>	不等於	BOOLEAN
||	串連	VARBYTE
+	串連	VARBYTE
~	位元 not	VARBYTE
&	位元 and	VARBYTE
|	位元 or	VARBYTE
#	位元 xor	VARBYTE

範例

在下列範例中，'a'::VARBYTE 的值為 61，'b'::VARBYTE 的值為 62。:: 將字串轉換為 VARBYTE 資料類型。如需轉換資料類型的詳細資訊，請參閱 CAST。

若要使用 < 運算子比較 'a' 是否小於 'b'，請使用下列範例。

```
SELECT 'a'::VARBYTE < 'b'::VARBYTE AS less_than;

+-----------+
| less_than |
+-----------+
| true      |
+-----------+
```

若要使用 = 運算子來比較 'a' 是否等於 'b'，請使用下列範例。

```
SELECT 'a'::VARBYTE = 'b'::VARBYTE AS equal;
```

```
+-------+
| equal |
+-------+
| false |
+-------+
```

若要使用 || 運算子連接兩個二進位值，請使用下列範例。

```
SELECT 'a'::VARBYTE || 'b'::VARBYTE AS concat;

+--------+
| concat |
+--------+
|   6162 |
+--------+
```

若要使用 + 運算子連接兩個二進位值，請使用下列範例。

```
SELECT 'a'::VARBYTE + 'b'::VARBYTE AS concat;

+--------+
| concat |
+--------+
|   6162 |
+--------+
```

若要使用 FROM_VARBYTE 函數求反輸入二進位值的每一位，請使用下列範例。字串 'a' 評估為 01100001。如需詳細資訊，請參閱FROM_VARBYTE。

```
SELECT FROM_VARBYTE(~'a'::VARBYTE, 'binary');

+--------------+
| from_varbyte |
+--------------+
|     10011110 |
+--------------+
```

若要在兩個輸入二進位值上套用 & 運算子，請使用下列範例。字串 'a' 評估為 01100001 且 'b' 評估為 01100010。

```
SELECT FROM_VARBYTE('a'::VARBYTE & 'b'::VARBYTE, 'binary');
```

```
+--------------+
| from_varbyte |
+--------------+
|     01100000 |
+--------------+
```

FROM_HEX 函數

FROM_HEX 將十六進位轉換為二進位值。

語法

```
FROM_HEX(hex_string)
```

引數

hex_string

要轉換的 VARCHAR 或 TEXT 資料類型十六進位字串。格式必須是文字值。

傳回類型

VARBYTE

範例

若要將 '6162' 的十六進位表示轉換為二進位值，請使用下列範例。結果會自動顯示為二進位值的十六進位表示。

```
SELECT FROM_HEX('6162');

+----------+
| from_hex |
+----------+
|     6162 |
+----------+
```

FROM_VARBYTE 函數

FROM_VARBYTE 會將二進位值轉換成指定格式的字元字串。

語法

```
FROM_VARBYTE(binary_value, format)
```

引數

binary_value

VARBYTE 資料類型的二進位值。

format

傳回字元字串的格式。不區分大小寫的有效值為 hex、binary、utf8 (utf-8 和 utf_8) 和 base64。

傳回類型

VARCHAR

範例

若要將二進位值'ab' 轉換為十六進位，請使用下列範例。

```
SELECT FROM_VARBYTE('ab', 'hex');

+--------------+
| from_varbyte |
+--------------+
|         6162 |
+--------------+
```

若要傳回 '4d' 的二進位表示法，請使用下列範例。'4d' 的二進位表示法是字元字串 01001101。

```
SELECT FROM_VARBYTE(FROM_HEX('4d'), 'binary');

+--------------+
| from_varbyte |
+--------------+
|     01001101 |
+--------------+
```

GETBIT 函數

GETBIT 傳回指定索引處的二進位值的位元值。

語法

```
GETBIT(binary_value, index)
```

引數

binary_value

VARBYTE 資料類型的二進位值。

索引

傳回之二進位值中位元的索引編號。二進位值是從 0 開始的位元組，從最右邊的位元 (最低有效位元) 到最左邊的位元 (最高有效位元) 進行索引。

傳回類型

INTEGER

範例

若要傳回二進位值 from_hex('4d') 索引 2 處的位元，請使用以下範例。'4d' 的二進位表示法是 01001101。

```
SELECT GETBIT(FROM_HEX('4d'), 2);

+--------+
| getbit |
+--------+
|      1 |
+--------+
```

若要在 from_hex('4d') 傳回的二進位值的八個索引位置傳回位元，請使用下列範例。'4d' 的二進位表示法是 01001101。

```
SELECT GETBIT(FROM_HEX('4d'), 7), GETBIT(FROM_HEX('4d'), 6),
```

```
  GETBIT(FROM_HEX('4d'), 5), GETBIT(FROM_HEX('4d'), 4),
  GETBIT(FROM_HEX('4d'), 3), GETBIT(FROM_HEX('4d'), 2),
  GETBIT(FROM_HEX('4d'), 1), GETBIT(FROM_HEX('4d'), 0);

+--------+--------+--------+--------+--------+--------+--------+--------+
| getbit | getbit | getbit | getbit | getbit | getbit | getbit | getbit |
+--------+--------+--------+--------+--------+--------+--------+--------+
|      0 |      1 |      0 |      0 |      1 |      1 |      0 |      1 |
+--------+--------+--------+--------+--------+--------+--------+--------+
```

TO_HEX 函數

TO_HEX 將數字或二進位值轉換為十六進位表示法。

語法

```
TO_HEX(value)
```

引數

值

要轉換的數字或二進位值 (VARBYTE)。

傳回類型

VARCHAR

範例

若要將數字轉換為其十六進位表示法，請使用下列範例。

```
SELECT TO_HEX(2147676847);

+----------+
|  to_hex  |
+----------+
| 8002f2af |
+----------+
```

若要將 'abc' 的 VARBYTE 表示轉換為十六進位數字，請使用下列範例。

```
SELECT TO_HEX('abc'::VARBYTE);

+--------+
| to_hex |
+--------+
| 616263 |
+--------+
```

若要建立資料表、將 'abc' 的 VARBYTE 表示法插入至十六進位數字，然後選取含值的欄，請使用下列範例。

```
CREATE TABLE t (vc VARCHAR);
INSERT INTO t SELECT TO_HEX('abc'::VARBYTE);
SELECT vc FROM t;

+--------+
|   vc   |
+--------+
| 616263 |
+--------+
```

若要顯示將 VARBYTE 值轉換為 VARCHAR 時格式為 UTF-8，請使用下列範例。

```
CREATE TABLE t (vc VARCHAR);
INSERT INTO t SELECT 'abc'::VARBYTE::VARCHAR;

SELECT vc FROM t;

+-----+
| vc  |
+-----+
| abc |
+-----+
```

TO_VARBYTE 函數

TO_VARBYTE 會將指定格式的字串轉換為二進位值。

語法

```
TO_VARBYTE(string, format)
```

引數

string

CHAR 或 VARCHAR 字串。

format

輸入字串的格式。不區分大小寫的有效值為 hex、binary、utf8 (utf-8 和 utf_8) 和 base64。

傳回類型

VARBYTE

範例

若要將十六進位 6162 轉換為二進位值，請使用下列範例。結果會自動顯示為二進位值的十六進位表示。

```
SELECT TO_VARBYTE('6162', 'hex');

+------------+
| to_varbyte |
+------------+
|       6162 |
+------------+
```

若要傳回 4d 的二進位表示法，請使用下列範例。'4d' 的二進位表示是 01001101。

```
SELECT TO_VARBYTE('01001101', 'binary');

+------------+
| to_varbyte |
+------------+
|         4d |
+------------+
```

若要將 UTF-8 的 'a' 字串轉換為二進位值，請使用下列範例。結果會自動顯示為二進位值的十六進位表示。

```
SELECT TO_VARBYTE('a', 'utf8');
```

```
+------------+
| to_varbyte |
+------------+
|         61 |
+------------+
```

若要將十六進位的 '4' 字串轉換為二進位值，請使用下列範例。如果十六進位字串長度是奇數，則在前面加上 0 以形成有效的十六進位數字。

```
SELECT TO_VARBYTE('4', 'hex');

+------------+
| to_varbyte |
+------------+
|         04 |
+------------+
```

範圍函數

使用範圍函數，您可以更有效地建立分析業務查詢。範圍函數對結果集的一個分割區或「視窗」執行運算，然後針對該視窗中的每一列傳回一個值。反之，非視窗函數對結果集的每一列執行計算。不同於彙總結果列的群組函數，範圍函數會保留運算式中的所有列。

傳回的值是利用該視窗中列集的值來計算。對於資料表的每一列，視窗會定義用於計算其他屬性的列集。視窗是以視窗規格 (OVER 子句) 並根據三個主要概念來定義：

- 視窗分割，其會形成列群組 (PARTITION 子句)
- 視窗排序，定義每一個分割區內列的順序或序列 (ORDER BY 子句)
- 視窗框，相對於每一列來定義，以進一步限制列組 (ROWS 規格)

範圍函數是查詢中最後執行的一組運算 (最後的 ORDER BY 子句除外)。所有聯結和所有 WHERE、GROUP BY 及 HAVING 子句都在範圍函數處理之前完成。因此，範圍函數只能出現在 select 清單或 ORDER BY 子句中。您可以在具有不同窗框子句的單一查詢中使用多個範圍函數。您也可以在其他純量運算式 (例如 CASE) 中使用範圍函數。

範圍函數語法摘要

範圍函數遵循標準語法，如下所示。

```
function (expression) OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list [ frame_clause ] ] )
```

其中，function 是本節所述其中一個函數。

expr_list 如下。

```
expression | column_name [, expr_list ]
```

order_list 如下。

```
expression | column_name [ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[, order_list ]
```

frame_clause 如下。

```
ROWS
{ UNBOUNDED PRECEDING | unsigned_value PRECEDING | CURRENT ROW } |

{ BETWEEN
{ UNBOUNDED PRECEDING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW}
AND
{ UNBOUNDED FOLLOWING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW }}
```

引數

函數

範圍函數。如需詳細資訊，請參閱個別函數描述。

OVER

此子句定義視窗規格。OVER 是範圍函數的必要子句，用於區分範圍函數和其他 SQL 函數。

PARTITION BY expr_list

(選用) PARTITION BY 子句將結果集細分為分割區，很像 GROUP BY 子句。如果有分割區子句，則會對每一個分割區的列來計算函數。如果未指定分割區子句，則單一分割區包含整個資料表，且會針對這整個資料表來計算函數。

排名函數 DENSE_RANK、NTILE、RANK 及 ROW_NUMBER 需要整體比較結果集的所有列。使用 PARTITION BY 子句時，查詢最佳化工具可以根據分割區將工作負載分散至多個配量，以平行執行每一個彙總。如果沒有 PARTITION BY 子句，則必須在單一配量上循序執行彙總步驟，這可能對效能造成嚴重的負面影響，尤其對於大型叢集。

Amazon Redshift 不支援 PARTITION BY 子句中的字串常值。

ORDER BY order_list

(選用) 範圍函數會套用至每一個分割區內根據 ORDER BY 中的順序規格所排序的列。此 ORDER BY 子句不同於且完全無關於 frame_clause 中的 ORDER BY 子句。使用 ORDER BY 子句可以不搭配 PARTITION BY 子句。

對於排名函數，ORDER BY 子句可辨識排名值的量值。對於彙總函數，在為每一個窗框計算彙總函數之前，分割的列必須排序。如需範圍函數的詳細資訊，請參閱範圍函數。

順序清單中需要欄識別碼或可評估為欄識別碼的欄表達式。常數或常數表達式都不能用來替代欄名。

NULLS 值自成一組，根據 NULLS FIRST 或 NULLS LAST 選項來排序和排名。根據預設，依 ASC 順序排序時，NULL 值排在最後面，而依 DESC 順序排序時，則排在最前面。

Amazon Redshift 不支援 ORDER BY 子句中的字串常值。

如果省略 ORDER BY 子句，則列的順序不確定。

> **Note**
>
> 在任何平行系統中，例如 Amazon Redshift，當 ORDER BY 子句無法產生唯一且完全的資料排序時，列的順序不確定。也就是說，如果 ORDER BY 運算式產生重複值 (局部排序)，則那些列的傳回順序可能隨著每一次執行 Amazon Redshift 而有所不同。於是，範圍函數可能傳回非預期或不一致的結果。如需詳細資訊，請參閱 範圍函數的資料唯一排序。

column_name

分割或排序所依據的欄名。

ASC | DESC

此選項會定義表達式的排序順序，如下所示：

- ASC：遞增 (例如，數值從低到高，字元字串 'A' 到 'Z')。若未指定選項，資料會預設為遞增排序。

- DESC：遞減 (數值從高到低，字串 'Z' 到 'A')。

NULLS FIRST | NULLS LAST

這些選項指定 NULLS 應該排序在最前 (在非 Null 值之前) 或排序在最後 (在非 Null 值之後)。根據預設，NULLS 在 ASC 排序中排序和排名最後，而在 DESC 排序中排序和排名最前。

frame_clause

對於彙總函數，使用 ORDER BY 時，窗框子句會進一步調整函數視窗中的一個列集。它可讓您在排序的結果內包含或排除資料列組。窗框子句包含 ROWS 關鍵字和相關的指定元。

窗框子句不適用於排名函數。此外，當彙總函數的 OVER 子句中未使用 ORDER BY 子句時，不需要使用窗框子句。如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。

未指定 ORDER BY 子句時，隱含的窗框無邊界：相當於 ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING。

ROWS

此子句定義視窗框作法是指定相對於目前列的實體位移。

此子句指定目前視窗或分割區中的列，以便與目前列的值結合。此子句使用引數來指定列位置，可能在目前列之前或之後。所有視窗框都以目前列為參考點。隨著視窗框在分割區中向前滑動，每一列會輪流變成目前列。

窗框可能是一組簡單的列，最遠到達且包含目前列。

```
{UNBOUNDED PRECEDING | offset PRECEDING | CURRENT ROW}
```

也可能是兩個邊界之間的一個列集。

```
BETWEEN
{ UNBOUNDED PRECEDING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }
AND
{ UNBOUNDED FOLLOWING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }
```

UNBOUNDED PRECEDING 表示視窗從分割區的第一列開始；offset PRECEDING 表示視窗從目前列之前相當於 offset 值的列數開始。UNBOUNDED PRECEDING 是預設值。

CURRENT ROW 表示視窗在目前列開始或結束。

UNBOUNDED FOLLOWING 表示視窗在分割區的最後一列結束；offset FOLLOWING 表示視窗在目前列之後相當於 offset 值的列數結束。

offset 表示目前列之前或之後的實體列數。在此案例中，offset 必須是評估為正數值的常數。例如，5 FOLLOWING 會在目前列之後的 5 列結束窗框。

未指定 BETWEEN 時，窗框會隱含地以目前列為邊界。例如，ROWS 5 PRECEDING 等於 ROWS BETWEEN 5 PRECEDING AND CURRENT ROW。此外，ROWS UNBOUNDED FOLLOWING 等於 ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING。

> **Note**
>
> 您不能指定開始邊界大於結束邊界的窗框。例如，您不能指定下列任何窗框：
>
> ```
> between 5 following and 5 preceding
> between current row and 2 preceding
> between 3 following and current row
> ```

範圍函數的資料唯一排序

如果範圍函數的 ORDER BY 子句無法產生唯一且完全的資料排序時，列的順序不確定。如果 ORDER BY 運算式產生重複值 (局部排序)，則這些行的傳回順序在多次執行中可能會有所不同。在這種情況下，範圍函數也可能傳回非預期或不一致的結果。

例如，以下查詢會在多次執行中傳回不同的結果。發生這些不同的結果是因為 order by dateid 不會為 SUM 範圍函數產生唯一的資料排序。

```
select dateid, pricepaid,
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid |   sumpaid
--------+-----------+-------------
1827 |   1730.00 |     1730.00
1827 |    708.00 |     2438.00
1827 |    234.00 |     2672.00
...

select dateid, pricepaid,
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;
```

```
dateid | pricepaid |   sumpaid
--------+-----------+-------------
1827 |    234.00 |      234.00
1827 |    472.00 |      706.00
1827 |    347.00 |     1053.00
...
```

在此情況下，將第二個 ORDER BY 欄新增至範圍函數可能會解決問題。

```
select dateid, pricepaid,
sum(pricepaid) over(order by dateid, pricepaid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid | sumpaid
--------+-----------+---------
1827 |    234.00 |  234.00
1827 |    337.00 |  571.00
1827 |    347.00 |  918.00
...
```

支援的函數

Amazon Redshift 支援兩種範圍函數：彙總和排名。

以下是支援的彙總函數：

- AVG 範圍函數
- COUNT 範圍函數
- CUME_DIST 範圍函數
- DENSE_RANK 範圍函數
- FIRST_VALUE 範圍函數
- LAG 範圍函數
- LAST_VALUE 範圍函數
- LEAD 範圍函數
- LISTAGG 範圍函數
- MAX 範圍函數

- MEDIAN 範圍函數
- MIN 範圍函數
- NTH_VALUE 範圍函數
- PERCENTILE_CONT 範圍函數
- PERCENTILE_DISC 範圍函數
- RATIO_TO_REPORT 範圍函數
- STDDEV_SAMP 和 STDDEV_POP 範圍函數 (STDDEV_SAMP 和 STDDEV 是同義詞)
- SUM 範圍函數
- VAR_SAMP 和 VAR_POP 範圍函數 (VAR_SAMP 和 VARIANCE 是同義詞)

以下是支援的排名函數：

- DENSE_RANK 範圍函數
- NTILE 範圍函數
- PERCENT_RANK 範圍函數
- RANK 範圍函數
- ROW_NUMBER 範圍函數

範圍函數範例的範例資料表

您可以在每個函數說明中找到特定的範圍函數範例。部分範例會使用名為 WINSALES 的資料表，其中包含 11 個資料列，如下所示。

SALESID	DATEID	SELLERID	BUYERID	QTY	QTY_SHIPPED
30001	8/2/2003	3	B	10	10
10001	12/24/2003	1	C	10	10
10005	12/24/2003	1	A	30	
40001	1/9/2004	4	A	40	
10006	1/18/2004	1	C	10	

SALESID	DATEID	SELLERID	BUYERID	QTY	QTY_SHIPPED
20001	2/12/2004	2	B	20	20
40005	2/12/2004	4	A	10	10
20002	2/16/2004	2	C	20	20
30003	4/18/2004	3	B	15	
30004	4/18/2004	3	B	20	
30007	9/7/2004	3	C	30	

下列指令碼建立並填入範例 WINSALES 資料表。

```
CREATE TABLE winsales(
  salesid int,
  dateid date,
  sellerid int,
  buyerid char(10),
  qty int,
  qty_shipped int);

INSERT INTO winsales VALUES
  (30001, '8/2/2003', 3, 'b', 10, 10),
  (10001, '12/24/2003', 1, 'c', 10, 10),
  (10005, '12/24/2003', 1, 'a', 30, null),
  (40001, '1/9/2004', 4, 'a', 40, null),
  (10006, '1/18/2004', 1, 'c', 10, null),
  (20001, '2/12/2004', 2, 'b', 20, 20),
  (40005, '2/12/2004', 4, 'a', 10, 10),
  (20002, '2/16/2004', 2, 'c', 20, 20),
  (30003, '4/18/2004', 3, 'b', 15, null),
  (30004, '4/18/2004', 3, 'b', 20, null),
  (30007, '9/7/2004', 3, 'c', 30, null);
```

AVG 範圍函數

AVG 範圍函數傳回輸入表達式值的平均值 (算術平均數)。AVG 函數處理數值，且忽略 NULL 值。

語法

```
AVG ( [ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

如果指定引數 ALL，則函數在計數時會保留表達式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句區分視窗彙總函數和正常組彙總函數。

PARTITION BY expr_list

以一或多個表達式定義 AVG 函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

資料類型

AVG 函數支援的引數類型包括 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL 及 DOUBLE PRECISION。

AVG 函數支援的傳回類型如下：

- BIGINT 代表 SMALLINT 或 INTEGER 引數
- NUMERIC 代表 BIGINT 引數
- DOUBLE PRECISION 代表浮點數引數

範例

下列範例會依日期計算銷售數量的移動平均數，依日期 ID 和銷售 ID 排序結果：

```
select salesid, dateid, sellerid, qty,
avg(qty) over
(order by dateid, salesid rows unbounded preceding) as avg
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | avg
---------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  10
10005 | 2003-12-24 |        1 |  30 |  16
40001 | 2004-01-09 |        4 |  40 |  22
10006 | 2004-01-18 |        1 |  10 |  20
20001 | 2004-02-12 |        2 |  20 |  20
40005 | 2004-02-12 |        4 |  10 |  18
20002 | 2004-02-16 |        2 |  20 |  18
30003 | 2004-04-18 |        3 |  15 |  18
30004 | 2004-04-18 |        3 |  20 |  18
30007 | 2004-09-07 |        3 |  30 |  19
(11 rows)
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

COUNT 範圍函數

COUNT 範圍函數會計算表達式所定義的列數。

COUNT 函數有兩種版本。COUNT(*) 計算目標資料表中的所有列數，而不論是否包含 Null。COUNT (表達式) 計算特定欄或表達式中不含 NULL 值的列數。

語法

```
COUNT ( * | [ ALL ] expression) OVER
(
```

```
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

如果指定引數 ALL，則函數在計數時會保留表達式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句區分視窗彙總函數和正常組彙總函數。

PARTITION BY expr_list

以一或多個表達式定義 COUNT 函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

資料類型

COUNT 函數支援所有引數資料類型。

COUNT 函數支援的傳回類型為 BIGINT。

範例

下列範例顯示從資料視窗開頭的所有列的銷售 ID、數量和計數：

```
select salesid, qty,
count(*) over (order by salesid rows unbounded preceding) as count
```

```
from winsales
order by salesid;

salesid | qty | count
---------+-----+-----
10001 |  10 |   1
10005 |  30 |   2
10006 |  10 |   3
20001 |  20 |   4
20002 |  20 |   5
30001 |  10 |   6
30003 |  15 |   7
30004 |  20 |   8
30007 |  30 |   9
40001 |  40 |   10
40005 |  10 |   11
(11 rows)
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

下列範例顯示如何從資料視窗開頭計算非 null 列的銷售 ID、數量和計數。(在 WINSALES 資料表中，QTY_SHIPPED 欄包含一些 NULL。)

```
select salesid, qty, qty_shipped,
count(qty_shipped)
over (order by salesid rows unbounded preceding) as count
from winsales
order by salesid;

salesid | qty | qty_shipped | count
---------+-----+-------------+-------
10001 |  10 |          10 |   1
10005 |  30 |             |   1
10006 |  10 |             |   1
20001 |  20 |          20 |   2
20002 |  20 |          20 |   3
30001 |  10 |          10 |   4
30003 |  15 |             |   4
30004 |  20 |             |   4
30007 |  30 |             |   4
40001 |  40 |             |   4
40005 |  10 |          10 |   5
(11 rows)
```

CUME_DIST 範圍函數

計算視窗或分割區內值的累積分佈。假定為遞增排序，使用此公式來決定累積分佈：

```
count of rows with values <= x / count of rows in the window or partition
```

其中，x 等於 ORDER BY 子句所指定欄之目前列中的值。以下資料集示範此公式的使用：

```
Row# Value   Calculation    CUME_DIST
1        2500    (1)/(5)    0.2
2        2600    (2)/(5)    0.4
3        2800    (3)/(5)    0.6
4        2900    (4)/(5)    0.8
5        3100    (5)/(5)    1.0
```

傳回值範圍是 >0 至 1 (含)。

語法

```
CUME_DIST ()
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)
```

引數

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗框架規格。

PARTITION BY partition_expression

選用。此表達式針對 OVER 子句中的每一個群組，設定記錄範圍。

ORDER BY order_list

要計算累積分佈的表達式。表達式必須為數值資料類型，或可隱含地轉換為數值資料類型。如果省略 ORDER BY，所有列的傳回值為 1。

如果 ORDER BY 未產生唯一排序，則列的順序不確定。如需詳細資訊，請參閱 範圍函數的資料唯一排序。

傳回類型

FLOAT8

範例

以下範例計算每一個賣方的數量累積分佈：

```
select sellerid, qty, cume_dist()
over (partition by sellerid order by qty)
from winsales;

sellerid   qty    cume_dist
------------------------------------------------
1          10.00    0.33
1          10.64    0.67
1          30.37    1
3          10.04    0.25
3          15.15    0.5
3          20.75    0.75
3          30.55    1
2          20.09    0.5
2          20.12    1
4          10.12    0.5
4          40.23    1
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

DENSE_RANK 範圍函數

DENSE_RANK 範圍函數根據 OVER 子句中的 ORDER BY 表達式，決定一組值之中某個值的排名。如果有選用的 PARTITION BY 子句，則會重設每一組列的排名。在排名準則中有相等值的列獲得相同排名。DENSE_RANK 函數有一方面不同於 RANK：如果兩列以上繫結在一起，則排名值的序列中沒有間隙。例如，假設兩列都排名 1，則下一個排名為 2。

在相同查詢中，排名函數可以搭配不同的 PARTITION BY 和 ORDER BY 子句。

語法

```
DENSE_RANK() OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list ]
```

```
)
```

引數

( )

此函數不接受引數，但需要空括號。

OVER

DENSE_RANK 函數的視窗子句。

PARTITION BY expr_list

(選用) 一或多個用於定義視窗的運算式。

ORDER BY order_list

(選用) 排名值所根據的運算式。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。如果省略 ORDER BY，所有列的傳回值為 1。

如果 ORDER BY 未產生唯一排序，則列的順序不確定。如需詳細資訊，請參閱 範圍函數的資料唯一排序。

傳回類型

INTEGER

範例

下列範例使用範圍函數的範例資料表。如需詳細資訊，請參閱 範圍函數範例的範例資料表。

以下範例依銷售數量排序資料表，並將密集排名和一般排名指派給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, qty,
DENSE_RANK() OVER(ORDER BY qty DESC) AS d_rnk,
RANK() OVER(ORDER BY qty DESC) AS rnk
FROM winsales
ORDER BY 2,1;

+---------+-----+-------+-----+
| salesid | qty | d_rnk | rnk |
+---------+-----+-------+-----+
```

```
|   10001 |  10 |     5 |   8 |
|   10006 |  10 |     5 |   8 |
|   30001 |  10 |     5 |   8 |
|   40005 |  10 |     5 |   8 |
|   30003 |  15 |     4 |   7 |
|   20001 |  20 |     3 |   4 |
|   20002 |  20 |     3 |   4 |
|   30004 |  20 |     3 |   4 |
|   10005 |  30 |     2 |   2 |
|   30007 |  30 |     2 |   2 |
|   40001 |  40 |     1 |   1 |
+---------+-----+-------+-----+
```

在相同查詢中同時使用 DENSE_RANK 和 RANK 函數時，請注意指派給相同列集的排名差異。

下列範例會依 sellerid 分割資料表，並依數量排序每一個分割區，然後指派密集排名給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, sellerid, qty,
DENSE_RANK() OVER(PARTITION BY sellerid ORDER BY qty DESC) AS d_rnk
FROM winsales
ORDER BY 2,3,1;

+---------+----------+-----+-------+
| salesid | sellerid | qty | d_rnk |
+---------+----------+-----+-------+
|   10001 |        1 |  10 |     2 |
|   10006 |        1 |  10 |     2 |
|   10005 |        1 |  30 |     1 |
|   20001 |        2 |  20 |     1 |
|   20002 |        2 |  20 |     1 |
|   30001 |        3 |  10 |     4 |
|   30003 |        3 |  15 |     3 |
|   30004 |        3 |  20 |     2 |
|   30007 |        3 |  30 |     1 |
|   40005 |        4 |  10 |     2 |
|   40001 |        4 |  40 |     1 |
+---------+----------+-----+-------+
```

若要成功使用最後範例，請使用下列命令將資料列插入 WINSALES 資料表。此列與另一列具有相同的 buyid、sellerid 和 qtysold。這將導致上一個範例中的兩列並列，因此將顯示 DENSE_RANK 和 RANK 函數之間的差異。

```
INSERT INTO winsales VALUES(30009, '2/2/2003', 3, 'b', 20, NULL);
```

下列範例會依 buyerid 和 sellerid 分割資料表，並依數量排序每一個分割區，然後同時指派密集排名和一般排名給每一列。(選用) 套用範圍函數之後對結果進行排序。

```
SELECT salesid, sellerid, qty, buyerid,
DENSE_RANK() OVER(PARTITION BY buyerid, sellerid ORDER BY qty DESC) AS d_rnk,
RANK() OVER (PARTITION BY buyerid, sellerid ORDER BY qty DESC) AS rnk
FROM winsales
ORDER BY rnk;

+---------+----------+-----+---------+-------+-----+
| salesid | sellerid | qty | buyerid | d_rnk | rnk |
+---------+----------+-----+---------+-------+-----+
|   20001 |        2 |  20 | b       |     1 |   1 |
|   30007 |        3 |  30 | c       |     1 |   1 |
|   10006 |        1 |  10 | c       |     1 |   1 |
|   10005 |        1 |  30 | a       |     1 |   1 |
|   20002 |        2 |  20 | c       |     1 |   1 |
|   30009 |        3 |  20 | b       |     1 |   1 |
|   40001 |        4 |  40 | a       |     1 |   1 |
|   30004 |        3 |  20 | b       |     1 |   1 |
|   10001 |        1 |  10 | c       |     1 |   1 |
|   40005 |        4 |  10 | a       |     2 |   2 |
|   30003 |        3 |  15 | b       |     2 |   3 |
|   30001 |        3 |  10 | b       |     3 |   4 |
+---------+----------+-----+---------+-------+-----+
```

FIRST_VALUE 範圍函數

在一組已排序的列中，FIRST_VALUE 會針對視窗框中的第一列，傳回指定之表達式的值。

如需有關選取視窗框中最後一列的資訊，請參閱LAST_VALUE 範圍函數。

語法

```
FIRST_VALUE( expression )[ IGNORE NULLS | RESPECT NULLS ]
OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

引數

運算式

函數運算的目標欄或表達式。

IGNORE NULLS

此選項與 FIRST_VALUE 一起使用時，函數會傳回窗框中第一個非 NULL (或如果所有值都是 NULL，則為 NULL) 的值。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。如果您不指定 IGNORE NULLS，則預設支援 RESPECT NULLS。

OVER

引進函數的視窗子句。

PARTITION BY expr_list

以一或多個表達式定義函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY 子句，ORDER BY 會排序整個資料表。如果您指定 ORDER BY 子句，則還必須指定 frame_clause。

FIRST_VALUE 函數的結果取決於資料的排序。在下列情況中，結果不確定：

- 未指定 ORDER BY 子句，且分割區包含一個表達式的兩個不同值
- 表達式評估為不同值，而這些值對應於 ORDER BY 清單中的相同值。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果中包含或排除資料列組，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

傳回類型

這些函數支援使用基本 Amazon Redshift 資料類型的運算式。傳回類型與運算式的資料類型相同。

範例

下列範例會使用範例 TICKIT 資料中的 VENUE 表格。如需詳細資訊，請參閱 範本資料庫。

下列範例傳回 VENUE 資料表中每個會場的座位容量，且結果依容量排序 (高到低)。會使用 FIRST_VALUE 函數來選取與窗框之第一列對應的會場名稱：在此案例中，即座位數最多的那一列。結果依州分割，所以當 VENUESTATE 值變更時，就會選取新的第一個值。視窗框無界限，對於每一個分割區的第一列，選取的第一個值都相同。

以加利佛尼亞來說，Qualcomm Stadium 的座位數最多 (70561)，因此，對於 CA 分割區中的所有列，此名稱是第一個值。

```
select venuestate, venueseats, venuename,
first_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;

venuestate | venueseats |           venuename            |         first_value
-----------+------------+--------------------------------
+------------------------------
CA         |      70561 | Qualcomm Stadium               | Qualcomm Stadium
CA         |      69843 | Monster Park                   | Qualcomm Stadium
CA         |      63026 | McAfee Coliseum                | Qualcomm Stadium
CA         |      56000 | Dodger Stadium                 | Qualcomm Stadium
CA         |      45050 | Angel Stadium of Anaheim       | Qualcomm Stadium
CA         |      42445 | PETCO Park                     | Qualcomm Stadium
CA         |      41503 | AT&T Park                      | Qualcomm Stadium
CA         |      22000 | Shoreline Amphitheatre         | Qualcomm Stadium
CO         |      76125 | INVESCO Field                  | INVESCO Field
CO         |      50445 | Coors Field                    | INVESCO Field
DC         |      41888 | Nationals Park                 | Nationals Park
FL         |      74916 | Dolphin Stadium                | Dolphin Stadium
FL         |      73800 | Jacksonville Municipal Stadium | Dolphin Stadium
FL         |      65647 | Raymond James Stadium          | Dolphin Stadium
FL         |      36048 | Tropicana Field                | Dolphin Stadium
...
```

下列範例顯示使用 IGNORE NULLS 選項，並依賴將新的一列新增至 VENUE 資料表：

```
insert into venue values(2000,null,'Stanford','CA',90000);
```

這個新列的 VENUENAME 欄包含 NULL 值。現在，重複本節稍早所示的 FIRST_VALUE 查詢：

```
select venuestate, venueseats, venuename,
first_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;

venuestate | venueseats |         venuename          | first_value
-----------+------------+----------------------------+-------------
CA         |      90000 | NULL                       | NULL
CA         |      70561 | Qualcomm Stadium           | NULL
CA         |      69843 | Monster Park               | NULL
...
```

因為新列包含最高 VENUESEATS 值 (90000)，且其 VENUENAME 為 NULL，所以 FIRST_VALUE 函數對 CA 分割區傳回 Null。若要在函數評估中忽略像這樣的列，請將 IGNORE NULLS 選項新增至函數引數：

```
select venuestate, venueseats, venuename,
first_value(venuename) ignore nulls
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venuestate='CA')
order by venuestate;

venuestate | venueseats |         venuename          |   first_value
-----------+------------+----------------------------+-----------------
CA         |      90000 | NULL                       | Qualcomm Stadium
CA         |      70561 | Qualcomm Stadium           | Qualcomm Stadium
CA         |      69843 | Monster Park               | Qualcomm Stadium
...
```

LAG 範圍函數

LAG 範圍函數傳回分割區中目前列上方 (之前) 給定位移那一列的值。

語法

```
LAG (value_expr [, offset ])
[ IGNORE NULLS | RESPECT NULLS ]
```

```
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

引數

value_expr

函數運算的目標欄或表達式。

offset

選擇性參數，指定在目前列之前要傳回值的列數。位移可以是常數整數，或評估為整數的表達式。如果您不指定位移，Amazon Redshift 會使用 1 做為預設值。位移 0 表示目前列。

IGNORE NULLS

選擇性規格，指出 Amazon Redshift 在決定要使用的列時應該略過 Null 值。如果未列出 IGNORE NULLS，則會包含 Null 值。

> **Note**
>
> 您可以使用 NVL 或 COALESCE 表達式，將 Null 值換成另一個值。如需詳細資訊，請參閱 NVL 和 COALESCE 函數。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。如果您不指定 IGNORE NULLS，則預設支援 RESPECT NULLS。

OVER

指定視窗分割和排序。OVER 子句不能包含視窗框規格。

PARTITION BY window_partition

選擇性引數，針對 OVER 子句中的每一個群組，設定記錄範圍。

ORDER BY window_ordering

排序每一個分割區內的列。

LAG 範圍函數支援有使用任何 Amazon Redshift 資料類型的運算式。傳回類型與 value_expr 的類型相同。

範例

下列範例顯示銷售給買方 ID 為 3 之買方的門票數量，以及買方 3 購買門票的時間。為了比較買方 3 的每次銷售與前次銷售，查詢會傳回每次銷售的前次銷售數量。因為 2008/1/16 之前沒有購買，所以第一個先前銷售數量值為 Null：

```
select buyerid, saletime, qtysold,
lag(qtysold,1) over (order by buyerid, saletime) as prev_qtysold
from sales where buyerid = 3 order by buyerid, saletime;

buyerid |      saletime       | qtysold | prev_qtysold
---------+---------------------+---------+--------------
3 | 2008-01-16 01:06:09 |       1 |
3 | 2008-01-28 02:10:01 |       1 |            1
3 | 2008-03-12 10:39:53 |       1 |            1
3 | 2008-03-13 02:56:07 |       1 |            1
3 | 2008-03-29 08:21:39 |       2 |            1
3 | 2008-04-27 02:39:01 |       1 |            2
3 | 2008-08-16 07:04:37 |       2 |            1
3 | 2008-08-22 11:45:26 |       2 |            2
3 | 2008-09-12 09:11:25 |       1 |            2
3 | 2008-10-01 06:22:37 |       1 |            1
3 | 2008-10-20 01:55:51 |       2 |            1
3 | 2008-10-28 01:30:40 |       1 |            2
(12 rows)
```

LAST_VALUE 範圍函數

在一組已排序的列中，LAST_VALUE 函數針對窗框中的最後一列，傳回運算式的值。

如需有關選取框架中第一列的資訊，請參閱FIRST_VALUE 範圍函數。

語法

```
LAST_VALUE( expression )[ IGNORE NULLS | RESPECT NULLS ]
OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

引數

運算式

函數運算的目標欄或表達式。

IGNORE NULLS

函數會傳回窗框中非 NULL (或如果所有值都是 NULL，則為 NULL) 的最後一個值。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。如果您不指定 IGNORE NULLS，則預設支援 RESPECT NULLS。

OVER

引進函數的視窗子句。

PARTITION BY expr_list

以一或多個表達式定義函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY 子句，ORDER BY 會排序整個資料表。如果您指定 ORDER BY 子句，則還必須指定 frame_clause。

結果取決於資料的順序。在下列情況中，結果不確定：

- 未指定 ORDER BY 子句，且分割區包含一個表達式的兩個不同值
- 表達式評估為不同值，而這些值對應於 ORDER BY 清單中的相同值。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果中包含或排除資料列組，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

傳回類型

這些函數支援使用基本 Amazon Redshift 資料類型的運算式。傳回類型與運算式的資料類型相同。

範例

下列範例會使用範例 TICKIT 資料中的 VENUE 表格。如需詳細資訊，請參閱 範本資料庫。

下列範例傳回 VENUE 資料表中每個會場的座位容量，且結果依容量排序 (高到低)。LAST_VALUE 函數用於選取與窗框之最後一列對應的會場名稱：在此案例中，即座位數最少的那一列。結果依州分割，所以當 VENUESTATE 值變更時，就會選取新的最後一個值。視窗框無界限，對於每一個分割區的第一列，選取的最後一個值都相同。

以加利佛尼亞來說，分割區中的每一列列都傳回 Shoreline Amphitheatre，因為其座位數最少 (22000)。

```
select venuestate, venueseats, venuename,
last_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;

venuestate | venueseats |           venuename            |          last_value
-----------+------------+--------------------------------
+-----------------------------
CA         |      70561 | Qualcomm Stadium               | Shoreline Amphitheatre
CA         |      69843 | Monster Park                   | Shoreline Amphitheatre
CA         |      63026 | McAfee Coliseum                | Shoreline Amphitheatre
CA         |      56000 | Dodger Stadium                 | Shoreline Amphitheatre
CA         |      45050 | Angel Stadium of Anaheim       | Shoreline Amphitheatre
CA         |      42445 | PETCO Park                     | Shoreline Amphitheatre
CA         |      41503 | AT&T Park                      | Shoreline Amphitheatre
CA         |      22000 | Shoreline Amphitheatre         | Shoreline Amphitheatre
CO         |      76125 | INVESCO Field                  | Coors Field
CO         |      50445 | Coors Field                    | Coors Field
DC         |      41888 | Nationals Park                 | Nationals Park
FL         |      74916 | Dolphin Stadium                | Tropicana Field
FL         |      73800 | Jacksonville Municipal Stadium | Tropicana Field
FL         |      65647 | Raymond James Stadium          | Tropicana Field
FL         |      36048 | Tropicana Field                | Tropicana Field
...
```

LEAD 範圍函數

LEAD 範圍函數傳回分割區中目前列下方 (之後) 給定位移那一列的值。

語法

```
LEAD (value_expr [, offset ])
```

```
[ IGNORE NULLS | RESPECT NULLS ]
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

引數

value_expr

函數運算的目標欄或表達式。

offset

選擇性參數，指定在目前列下方要傳回值的列數。位移可以是常數整數，或評估為整數的表達式。如果您不指定位移，Amazon Redshift 會使用 1 做為預設值。位移 0 表示目前列。

IGNORE NULLS

選擇性規格，指出 Amazon Redshift 在決定要使用的列時應該略過 Null 值。如果未列出 IGNORE NULLS，則會包含 Null 值。

> **Note**
>
> 您可以使用 NVL 或 COALESCE 表達式，將 Null 值換成另一個值。如需詳細資訊，請參閱 NVL 和 COALESCE 函數。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。如果您不指定 IGNORE NULLS，則預設支援 RESPECT NULLS。

OVER

指定視窗分割和排序。OVER 子句不能包含視窗框規格。

PARTITION BY window_partition

選擇性引數，針對 OVER 子句中的每一個群組，設定記錄範圍。

ORDER BY window_ordering

排序每一個分割區內的列。

LEAD 範圍函數支援有使用任何 Amazon Redshift 資料類型的運算式。傳回類型與 value_expr 的類型相同。

範例

下列範例提供 SALES 資料表中於 2008 年 1 月 1 日和 2008 年 1 月 2 日售出門票之活動的佣金，以及對隨後銷售之門票銷售支付的佣金。下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 範本資料庫。

```
SELECT eventid, commission, saletime, LEAD(commission, 1) over ( ORDER BY saletime ) AS
 next_comm
FROM sales
WHERE saletime BETWEEN '2008-01-09 00:00:00' AND '2008-01-10 12:59:59'
LIMIT 10;

+---------+------------+---------------------+-----------+
| eventid | commission |      saletime       | next_comm |
+---------+------------+---------------------+-----------+
|    1664 |       13.2 | 2008-01-09 01:00:21 |      69.6 |
|     184 |       69.6 | 2008-01-09 01:00:36 |     116.1 |
|    6870 |      116.1 | 2008-01-09 01:02:37 |      11.1 |
|    3718 |       11.1 | 2008-01-09 01:05:19 |     205.5 |
|    6772 |      205.5 | 2008-01-09 01:14:04 |      38.4 |
|    3074 |       38.4 | 2008-01-09 01:26:50 |     209.4 |
|    5254 |      209.4 | 2008-01-09 01:29:16 |      26.4 |
|    3724 |       26.4 | 2008-01-09 01:40:09 |      57.6 |
|    5303 |       57.6 | 2008-01-09 01:40:21 |      51.6 |
|    3678 |       51.6 | 2008-01-09 01:42:54 |      43.8 |
+---------+------------+---------------------+-----------+
```

LISTAGG 範圍函數

對於查詢中的每一組，LISTAGG 範圍函數依據 ORDER BY 表達式來排序該組的列，然後將這些值串連成單一字串。

LISTAGG 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。如需詳細資訊，請參閱 查詢目錄資料表。

語法

```
LISTAGG( [DISTINCT] expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]
OVER ( [PARTITION BY partition_expression] )
```

引數

DISTINCT

(選用) 此子句在串連值之前會從指定的表達式中消除重複值。結尾空格會忽略，所以字串 'a' 和 'a ' 視為重複值。LISTAGG 會使用第一個遇到的值。如需詳細資訊，請參閱 多餘空格的意義。

aggregate_expression

任何有效表達式 (例如欄名)，用於提供要彙總的值。忽略 NULL 值和空字串。

delimiter

(選用) 用來區隔串連值的字串常數。預設值為 NULL。

WITHIN GROUP (ORDER BY order_list)

(選用) 此子句指定彙總值的排序順序。只有在 ORDER BY 提供唯一排序時才可確定。預設是彙總所有列並傳回單一值。

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗排序或視窗框規格。

PARTITION BY partition_expression

(選用) 針對 OVER 子句中的每一個群組，設定記錄範圍。

傳回值

VARCHAR(MAX)。如果結果集大於 VARCHAR 大小上限 (64K – 1，或 65535)，則 LISTAGG 會傳回下列錯誤：

```
Invalid operation: Result size exceeds LISTAGG limit
```

範例

下列範例使用 WINSALES 資料表。如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

下列範例傳回賣方 ID 清單，依賣方 ID 排序。

```
select listagg(sellerid)
within group (order by sellerid)
over() from winsales;

  listagg
------------
 11122333344
...
...
 11122333344
 11122333344
   (11 rows)
```

下列範例傳回買方 B 的賣方 ID 清單，依日期排序。

```
select listagg(sellerid)
within group (order by dateid)
over () as seller
from winsales
where buyerid = 'b' ;

  seller
---------
    3233
    3233
    3233
    3233
```

下列範例以逗號分隔清單傳回買方 B 的銷售日期。

```
select listagg(dateid,',')
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid  = 'b';

             dates
----------------------------------------
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
```

下列範例使用 DISTINCT 傳回買方 B 的唯一銷售日期清單。

```
select listagg(distinct dateid,',')
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid  = 'b';

           dates
-------------------------------
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
```

下列範例以逗號分隔清單傳回每個買方 ID 的銷售 ID。

```
select buyerid,
listagg(salesid,',')
within group (order by salesid)
over (partition by buyerid) as sales_id
from winsales
order by buyerid;

+---------+------------------------+
| buyerid |        sales_id        |
+---------+------------------------+
| a       | 10005,40001,40005      |
| a       | 10005,40001,40005      |
| a       | 10005,40001,40005      |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
+---------+------------------------+
```

MAX 範圍函數

MAX 範圍函數傳回輸入表達式值的最大值。MAX 函數處理數值，且忽略 NULL 值。

語法

```
MAX ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

如果指定引數 ALL，函數會保留表達式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

此子句指定彙總函數的視窗子句。OVER 子句區分視窗彙總函數和正常組彙總函數。

PARTITION BY expr_list

以一或多個表達式定義 MAX 函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

資料類型

接受任何資料類型做為輸入。傳回與 expression 相同的資料類型。

範例

下列範例顯示資料視窗開頭的銷售 ID、數量和最大數量：

```
select salesid, qty,
```

```
max(qty) over (order by salesid rows unbounded preceding) as max
from winsales
order by salesid;

salesid | qty | max
---------+-----+-----
10001 |  10 |  10
10005 |  30 |  30
10006 |  10 |  30
20001 |  20 |  30
20002 |  20 |  30
30001 |  10 |  30
30003 |  15 |  30
30004 |  20 |  30
30007 |  30 |  30
40001 |  40 |  40
40005 |  10 |  40
(11 rows)
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

下列範例顯示受限窗框中的 salesid、數量和最大數量：

```
select salesid, qty,
max(qty) over (order by salesid rows between 2 preceding and 1 preceding) as max
from winsales
order by salesid;

salesid | qty | max
---------+-----+-----
10001 |  10 |
10005 |  30 |  10
10006 |  10 |  30
20001 |  20 |  30
20002 |  20 |  20
30001 |  10 |  20
30003 |  15 |  20
30004 |  20 |  15
30007 |  30 |  20
40001 |  40 |  30
40005 |  10 |  40
(11 rows)
```

MEDIAN 範圍函數

計算視窗或分割區內值範圍的中位數。忽略範圍中的 NULL 值。

MEDIAN 是採用連續分佈模型的反向分佈函數。

MEDIAN 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。

語法

```
MEDIAN ( median_expression )
OVER ( [ PARTITION BY partition_expression ] )
```

引數

median_expression

此表達式 (例如欄名) 提供要決定中位數的值。表達式必須為數值或日期時間資料類型，或可隱含地轉換為這種資料類型。

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗排序或視窗框規格。

PARTITION BY partition_expression

選用。此表達式針對 OVER 子句中的每一個群組，設定記錄範圍。

資料類型

傳回類型取決於 median_expression 的資料類型。下表顯示每一個 median_expression 資料類型的傳回類型。

輸入類型	傳回類型
INT2、INT4、INT8、NUMERIC、DECIMAL	DECIMAL
FLOAT、DOUBLE	DOUBLE
DATE	DATE

使用須知

如果 median_expression 引數是以最大精確度 38 位數定義的 DECIMAL 資料類型，MEDIAN 可能會傳回不準確的結果或錯誤。如果 MEDIAN 函數的傳回值超過 38 位數，會將結果截斷為適合長度，導致精確度降低。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些情況，建議使用精確度較低的資料類型，或將 median_expression 引數轉換為較低精確度。

例如，搭配 DECIMAL 引數的 SUM 函數傳回的預設精確度為 38 位數。結果的小數位數和引數的小數位數相同。因此，例如，DECIMAL(5,2) 欄的 SUM 會傳回 DECIMAL(38,2) 資料類型。

下列範例在 MEDIAN 函數的 median_expression 引數中使用 SUM 函數。PRICEPAID 欄的資料類型是 DECIMAL (8,2)，所以 SUM 函數會傳回 DECIMAL(38,2)。

```
select salesid, sum(pricepaid), median(sum(pricepaid))
over() from sales where salesid < 10 group by salesid;
```

為了避免可能降低精確度或溢位錯誤，請將結果轉換為精確度較低的 DECIMAL 資料類型，如下列範例所示。

```
select salesid, sum(pricepaid), median(sum(pricepaid)::decimal(30,2))
over() from sales where salesid < 10 group by salesid;
```

範例

以下範例計算每一個賣方的銷售數量中位數：

```
select sellerid, qty, median(qty)
over (partition by sellerid)
from winsales
order by sellerid;


sellerid qty median
---------------------------
1  10 10.0
1  10 10.0
1  30 10.0
2  20 20.0
2  20 20.0
3  10 17.5
3  15 17.5
```

```
3  20 17.5
3  30 17.5
4  10 25.0
4  40 25.0
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

MIN 範圍函數

MIN 範圍函數傳回輸入表達式值的最小值。MIN 函數處理數值，且忽略 NULL 值。

語法

```
MIN ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

如果指定引數 ALL，函數會保留表達式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句區分視窗彙總函數和正常組彙總函數。

PARTITION BY expr_list

以一或多個表達式定義 MIN 函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

資料類型

接受任何資料類型做為輸入。傳回與 expression 相同的資料類型。

範例

下列範例顯示資料視窗開頭的銷售 ID、數量和最小數量：

```
select salesid, qty,
min(qty) over
(order by salesid rows unbounded preceding)
from winsales
order by salesid;

salesid | qty | min
---------+-----+-----
10001 |  10 |  10
10005 |  30 |  10
10006 |  10 |  10
20001 |  20 |  10
20002 |  20 |  10
30001 |  10 |  10
30003 |  15 |  10
30004 |  20 |  10
30007 |  30 |  10
40001 |  40 |  10
40005 |  10 |  10
(11 rows)
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

下列範例顯示受限窗框中的銷售 ID、數量和最小數量：

```
select salesid, qty,
min(qty) over
(order by salesid rows between 2 preceding and 1 preceding) as min
from winsales
order by salesid;

salesid | qty | min
---------+-----+-----
10001 |  10 |
10005 |  30 |  10
```

```
10006 |  10 |  10
20001 |  20 |  10
20002 |  20 |  10
30001 |  10 |  20
30003 |  15 |  10
30004 |  20 |  10
30007 |  30 |  15
40001 |  40 |  20
40005 |  10 |  30
(11 rows)
```

NTH_VALUE 範圍函數

NTH_VALUE 範圍函數會相對於視窗的第一列，傳回視窗框之指定列的表達式值。

語法

```
NTH_VALUE (expr, offset)
[ IGNORE NULLS | RESPECT NULLS ]
OVER
( [ PARTITION BY window_partition ]
[ ORDER BY window_ordering
                        frame_clause ] )
```

引數

expr

函數運算的目標欄或表達式。

offset

相對於視窗中的第一列，決定要傳回表達式的列號。offset 可以是常數或表達式，且必須為大於 0 的正整數。

IGNORE NULLS

選擇性規格，指出 Amazon Redshift 在決定要使用的列時應該略過 Null 值。如果未列出 IGNORE NULLS，則會包含 Null 值。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。如果您不指定 IGNORE NULLS，則預設支援 RESPECT NULLS。

OVER

指定視窗分割、排序及視窗框。

PARTITION BY window_partition

針對 OVER 子句中的每一個群組，設定記錄範圍。

ORDER BY window_ordering

排序每一個分割區內的列。如果省略 ORDER BY，則預設窗框包含分割區中的所有列。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果中包含或排除資料列組，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

NTH_VALUE 範圍函數支援有使用任何 Amazon Redshift 資料類型的運算式。傳回類型與 expr 的類型相同。

範例

下列範例顯示加利佛尼亞、佛羅里達及紐約的前三大會場的座位數，並對照這些州其他會場的座位數：

```
select venuestate, venuename, venueseats,
nth_value(venueseats, 3)
ignore nulls
over(partition by venuestate order by venueseats desc
rows between unbounded preceding and unbounded following)
as third_most_seats
from (select * from venue where venueseats > 0 and
venuestate in('CA', 'FL', 'NY'))
order by venuestate;

venuestate |           venuename            | venueseats | third_most_seats
-----------+--------------------------------+------------+------------------
CA         | Qualcomm Stadium               |      70561 |            63026
CA         | Monster Park                   |      69843 |            63026
CA         | McAfee Coliseum                |      63026 |            63026
CA         | Dodger Stadium                 |      56000 |            63026
CA         | Angel Stadium of Anaheim       |      45050 |            63026
CA         | PETCO Park                     |      42445 |            63026
CA         | AT&T Park                      |      41503 |            63026
```

```
CA         | Shoreline Amphitheatre         |      22000 |            63026
FL         | Dolphin Stadium                |      74916 |            65647
FL         | Jacksonville Municipal Stadium |      73800 |            65647
FL         | Raymond James Stadium          |      65647 |            65647
FL         | Tropicana Field                |      36048 |            65647
NY         | Ralph Wilson Stadium           |      73967 |            20000
NY         | Yankee Stadium                 |      52325 |            20000
NY         | Madison Square Garden          |      20000 |            20000
(15 rows)
```

NTILE 範圍函數

NTILE 範圍函數將分割區中排序的列劃分為指定數量的排名群組，且大小儘可能相等，然後傳回給定列所屬的群組。

語法

```
NTILE (expr)
OVER (
[ PARTITION BY expression_list ]
[ ORDER BY order_list ]
)
```

引數

expr

每一個分割區的排名群組數，且結果必須為整數值 (大於 0)。expr 引數必須不可為 Null。

OVER

用於指定視窗分割和排序的子句。OVER 子句不能包含視窗框規格。

PARTITION BY window_partition

選用。OVER 子句中每一個群組的記錄範圍。

ORDER BY window_ordering

選用。此表達式排序每一個分割區內的列。如果省略 ORDER BY 子句，則排名行為相同。

如果 ORDER BY 未產生唯一排序，則列的順序不確定。如需詳細資訊，請參閱 範圍函數的資料唯一排序。

傳回類型

BIGINT

範例

下列範例將 2008 年 8 月 26 日 Hamlet 門票的支付價格分成四個排名群組。結果集有 17 列，幾乎平均分散於排名 1 到 4：

```
select eventname, caldate, pricepaid, ntile(4)
over(order by pricepaid desc) from sales, event, date
where sales.eventid=event.eventid and event.dateid=date.dateid and eventname='Hamlet'
and caldate='2008-08-26'
order by 4;

eventname |  caldate   | pricepaid | ntile
----------+------------+-----------+-------
Hamlet    | 2008-08-26 |   1883.00 |     1
Hamlet    | 2008-08-26 |   1065.00 |     1
Hamlet    | 2008-08-26 |    589.00 |     1
Hamlet    | 2008-08-26 |    530.00 |     1
Hamlet    | 2008-08-26 |    472.00 |     1
Hamlet    | 2008-08-26 |    460.00 |     2
Hamlet    | 2008-08-26 |    355.00 |     2
Hamlet    | 2008-08-26 |    334.00 |     2
Hamlet    | 2008-08-26 |    296.00 |     2
Hamlet    | 2008-08-26 |    230.00 |     3
Hamlet    | 2008-08-26 |    216.00 |     3
Hamlet    | 2008-08-26 |    212.00 |     3
Hamlet    | 2008-08-26 |    106.00 |     3
Hamlet    | 2008-08-26 |    100.00 |     4
Hamlet    | 2008-08-26 |     94.00 |     4
Hamlet    | 2008-08-26 |     53.00 |     4
Hamlet    | 2008-08-26 |     25.00 |     4
(17 rows)
```

PERCENT_RANK 範圍函數

計算給定資料列的百分比排行。使用此公式決定百分比排名：

```
(x - 1) / (the number of rows in the window or partition - 1)
```

其中 x 是目前列的排名。以下資料集示範此公式的使用：

```
Row# Value Rank Calculation PERCENT_RANK
1 15 1 (1-1)/(7-1) 0.0000
2 20 2 (2-1)/(7-1) 0.1666
3 20 2 (2-1)/(7-1) 0.1666
4 20 2 (2-1)/(7-1) 0.1666
5 30 5 (5-1)/(7-1) 0.6666
6 30 5 (5-1)/(7-1) 0.6666
7 40 7 (7-1)/(7-1) 1.0000
```

傳回值範圍是 0 至 1 (含)。任何集的第一列的 PERCENT_RANK 為 0。

語法

```
PERCENT_RANK ()
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)
```

引數

( )

此函數不接受引數，但需要空括號。

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗框規格。

PARTITION BY partition_expression

選用。此表達式針對 OVER 子句中的每一個群組，設定記錄範圍。

ORDER BY order_list

選用。要計算百分比排名的表達式。表達式必須為數值資料類型，或可隱含地轉換為數值資料類型。如果省略 ORDER BY，所有列的傳回值為 0。

如果 ORDER BY 未產生唯一排序，則列的順序不確定。如需詳細資訊，請參閱 範圍函數的資料唯一排序。

傳回類型

FLOAT8

範例

以下範例計算每一個賣方的銷售數量百分比排名：

```
select sellerid, qty, percent_rank()
over (partition by sellerid order by qty)
from winsales;

sellerid qty  percent_rank
----------------------------------------
1  10.00  0.0
1  10.64  0.5
1  30.37  1.0
3  10.04  0.0
3  15.15  0.33
3  20.75  0.67
3  30.55  1.0
2  20.09  0.0
2  20.12  1.0
4  10.12  0.0
4  40.23  1.0
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

PERCENTILE_CONT 範圍函數

PERCENTILE_CONT 是採用連續分佈模型的反向分佈函數。它採用百分位數值和排序規格，且會傳回插入值，該值將根據排序規格落入給定的百分位數值。

PERCENTILE_CONT 在值排序後計算值之間的線性插值。此函數在列根據排序規格來排序後，使用彙總群組中的百分位數值 (P) 和非 Null 列數 (N) 來計算列號。此列號 (RN) 是根據公式 RN = (1+ (P*(N-1)) 來計算。彙總函數的最終結果是以列號 CRN = CEILING(RN) 到 FRN = FLOOR(RN) 各列的值之間的線性插值來計算。

最終結果如下。

如果 (CRN = FRN = RN)，則結果為 (value of expression from row at RN)

否則結果如下：

(CRN - RN) * (value of expression for row at FRN) + (RN - FRN) * (value of expression for row at CRN).

您只能在 OVER 子句中指定 PARTITION 子句。如果指定 PARTITION，則對於每一列，PERCENTILE_CONT 會傳回在給定分割區內的一組值之中落在指定百分位數的值。

PERCENTILE_CONT 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。

語法

```
PERCENTILE_CONT ( percentile )
WITHIN GROUP (ORDER BY expr)
OVER (  [ PARTITION BY expr_list ]  )
```

引數

percentile

介於 0 和 1 之間的數值常數。計算時會忽略 Null。

WITHIN GROUP (ORDER BY expr)

指定要排序和計算百分位數的數值或日期/時間值。

OVER

指定視窗分割。OVER 子句不能包含視窗排序或視窗框規格。

PARTITION BY expr

選擇性引數，針對 OVER 子句中的每一個群組，設定記錄範圍。

傳回值

傳回類型取決於 WITHIN GROUP 子句中 ORDER BY 表達式的資料類型。下表顯示每一個 ORDER BY 表達式資料類型的傳回類型。

輸入類型	傳回類型
INT2、INT4、INT8、NUMERIC、DECIMAL	DECIMAL
FLOAT、DOUBLE	DOUBLE
DATE	DATE

輸入類型	傳回類型
TIMESTAMP	TIMESTAMP

使用須知

如果 ORDER BY 表達式是以最大精確度 38 位數定義的 DECIMAL 資料類型，PERCENTILE_CONT 可能會傳回不準確的結果或錯誤。如果 PERCENTILE_CONT 函數的傳回值超過 38 位數，結果會截斷為適合長度，導致精確度降低。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些情況，建議使用精確度較低的資料類型，或將 ORDER BY 表達式轉換為較低精確度。

例如，搭配 DECIMAL 引數的 SUM 函數傳回的預設精確度為 38 位數。結果的小數位數和引數的小數位數相同。因此，例如，DECIMAL(5,2) 欄的 SUM 會傳回 DECIMAL(38,2) 資料類型。

下列範例在 PERCENTILE_CONT 函數的 ORDER BY 子句中使用 SUM 函數。PRICEPAID 欄的資料類型是 DECIMAL (8,2)，所以 SUM 函數會傳回 DECIMAL(38,2)。

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid) desc) over()
from sales where salesid < 10 group by salesid;
```

為了避免可能降低精確度或溢位錯誤，請將結果轉換為精確度較低的 DECIMAL 資料類型，如下列範例所示。

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid)::decimal(30,2) desc) over()
from sales where salesid < 10 group by salesid;
```

範例

下列範例使用 WINSALES 資料表。如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

```
select sellerid, qty, percentile_cont(0.5)
within group (order by qty)
over() as median from winsales;

 sellerid | qty | median
```

```
----------+-----+--------
        1 |  10 |   20.0
        1 |  10 |   20.0
        3 |  10 |   20.0
        4 |  10 |   20.0
        3 |  15 |   20.0
        2 |  20 |   20.0
        3 |  20 |   20.0
        2 |  20 |   20.0
        3 |  30 |   20.0
        1 |  30 |   20.0
        4 |  40 |   20.0
(11 rows)
```

```
select sellerid, qty, percentile_cont(0.5)
within group (order by qty)
over(partition by sellerid) as median from winsales;

 sellerid | qty | median
----------+-----+--------
        2 |  20 |   20.0
        2 |  20 |   20.0
        4 |  10 |   25.0
        4 |  40 |   25.0
        1 |  10 |   10.0
        1 |  10 |   10.0
        1 |  30 |   10.0
        3 |  10 |   17.5
        3 |  15 |   17.5
        3 |  20 |   17.5
        3 |  30 |   17.5
(11 rows)
```

以下範例計算華盛頓州之賣方門票銷售的 PERCENTILE_CONT 和 PERCENTILE_DISC。

```
SELECT sellerid, state, sum(qtysold*pricepaid) sales,
percentile_cont(0.6) within group (order by sum(qtysold*pricepaid::decimal(14,2) )
 desc) over(),
percentile_disc(0.6) within group (order by sum(qtysold*pricepaid::decimal(14,2) )
 desc) over()
from sales s, users u
where s.sellerid = u.userid and state = 'WA' and sellerid < 1000
group by sellerid, state;
```

```
 sellerid | state |  sales  | percentile_cont | percentile_disc
----------+-------+---------+-----------------+-----------------
      127 | WA    | 6076.00 |         2044.20 |         1531.00
      787 | WA    | 6035.00 |         2044.20 |         1531.00
      381 | WA    | 5881.00 |         2044.20 |         1531.00
      777 | WA    | 2814.00 |         2044.20 |         1531.00
       33 | WA    | 1531.00 |         2044.20 |         1531.00
      800 | WA    | 1476.00 |         2044.20 |         1531.00
        1 | WA    | 1177.00 |         2044.20 |         1531.00
(7 rows)
```

PERCENTILE_DISC 範圍函數

PERCENTILE_DISC 是採用離散分佈模型的反向分佈函數。它採用百分位數值和排序規格，且會傳回給定集裡的一個元素。

針對給定的百分位數值 P，PERCENTILE_DISC 排序 ORDER BY 子句中的表達式值，且傳回的值具有大於或等於 P 的最小累積分佈值 (根據相同的排序規格)。

您只能在 OVER 子句中指定 PARTITION 子句。

PERCENTILE_DISC 是僅限於運算節點的函數。如果查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表，此函數會傳回錯誤。

語法

```
PERCENTILE_DISC ( percentile )
WITHIN GROUP (ORDER BY expr)
OVER (  [ PARTITION BY expr_list ]  )
```

引數

percentile

介於 0 和 1 之間的數值常數。計算時會忽略 Null。

WITHIN GROUP (ORDER BY expr)

指定要排序和計算百分位數的數值或日期/時間值。

OVER

指定視窗分割。OVER 子句不能包含視窗排序或視窗框規格。

PARTITION BY expr

選擇性引數，針對 OVER 子句中的每一個群組，設定記錄範圍。

傳回值

資料類型與 WITHIN GROUP 子句中的 ORDER BY 表達式相同。

範例

下列範例使用 WINSALES 資料表。如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

```
SELECT sellerid, qty, PERCENTILE_DISC(0.5)
WITHIN GROUP (ORDER BY qty)
OVER() AS MEDIAN FROM winsales;

+----------+-----+--------+
| sellerid | qty | median |
+----------+-----+--------+
| 3        | 10  | 20     |
| 1        | 10  | 20     |
| 1        | 10  | 20     |
| 4        | 10  | 20     |
| 3        | 15  | 20     |
| 2        | 20  | 20     |
| 2        | 20  | 20     |
| 3        | 20  | 20     |
| 1        | 30  | 20     |
| 3        | 30  | 20     |
| 4        | 40  | 20     |
+----------+-----+--------+

SELECT sellerid, qty, PERCENTILE_DISC(0.5)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS MEDIAN FROM winsales;

+----------+-----+--------+
| sellerid | qty | median |
+----------+-----+--------+
| 4        | 10  | 10     |
| 4        | 40  | 10     |
| 3        | 10  | 15     |
```

```
| 3         | 15  | 15     |
| 3         | 20  | 15     |
| 3         | 30  | 15     |
| 2         | 20  | 20     |
| 2         | 20  | 20     |
| 1         | 10  | 10     |
| 1         | 10  | 10     |
| 1         | 30  | 10     |
+----------+-----+--------+
```

若要尋找依賣家 ID 分割時數量的 PERCENTILE_DISC(0.25) 和 PERCENTILE_DISC(0.75)，請使用下列範例。

```
SELECT sellerid, qty, PERCENTILE_DISC(0.25)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS quartile1 FROM winsales;

+----------+-----+-----------+
| sellerid | qty | quartile1 |
+----------+-----+-----------+
| 4         | 10  | 10        |
| 4         | 40  | 10        |
| 2         | 20  | 20        |
| 2         | 20  | 20        |
| 3         | 10  | 10        |
| 3         | 15  | 10        |
| 3         | 20  | 10        |
| 3         | 30  | 10        |
| 1         | 10  | 10        |
| 1         | 10  | 10        |
| 1         | 30  | 10        |
+----------+-----+-----------+

SELECT sellerid, qty, PERCENTILE_DISC(0.75)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS quartile3 FROM winsales;

+----------+-----+-----------+
| sellerid | qty | quartile3 |
+----------+-----+-----------+
| 3         | 10  | 20        |
| 3         | 15  | 20        |
| 3         | 20  | 20        |
```

```
| 3         | 30  | 20         |
| 4         | 10  | 40         |
| 4         | 40  | 40         |
| 2         | 20  | 20         |
| 2         | 20  | 20         |
| 1         | 10  | 30         |
| 1         | 10  | 30         |
| 1         | 30  | 30         |
+-----------+-----+------------+
```

RANK 範圍函數

RANK 範圍函數根據 OVER 子句中的 ORDER BY 表達式，決定一組值之中某個值的排名。如果有選用的 PARTITION BY 子句，則會重設每一組列的排名。在排名準則中有相等值的列獲得相同排名。Amazon Redshift 將綁定的行數添加到綁定的排名中以計算下一個排名，因此排名可能不是連續的數字。例如，假設兩列都排名 1，則下一個排名為 3。

RANK 函數有一方面不同於 DENSE_RANK 範圍函數：對於 DENSE_RANK，如果兩列以上繫結在一起，則排名值的序列中沒有間隙。例如，假設兩列都排名 1，則下一個排名為 2。

在相同查詢中，排名函數可以搭配不同的 PARTITION BY 和 ORDER BY 子句。

語法

```
RANK () OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list ]
)
```

引數

( )

此函數不接受引數，但需要空括號。

OVER

RANK 函數的視窗子句。

PARTITION BY expr_list

選用。一或多個用於定義視窗的表達式。

ORDER BY order_list

選用。定義排名值所根據的欄。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。如果省略 ORDER BY，所有列的傳回值為 1。

如果 ORDER BY 未產生唯一排序，則列的順序不確定。如需詳細資訊，請參閱 範圍函數的資料唯一排序。

傳回類型

INTEGER

範例

下列範例會依銷售數量排序資料表 (預設為遞增)，並將排名指派給每一列。最高排名的值為 1。套用範圍函數結果之後排序結果：

```
select salesid, qty,
rank() over (order by qty) as rnk
from winsales
order by 2,1;

salesid | qty | rnk
--------+-----+-----
10001 |  10 |  1
10006 |  10 |  1
30001 |  10 |  1
40005 |  10 |  1
30003 |  15 |  5
20001 |  20 |  6
20002 |  20 |  6
30004 |  20 |  6
10005 |  30 |  9
30007 |  30 |  9
40001 |  40 |  11
(11 rows)
```

請注意，此範例中的外圍 ORDER BY 子句包含第 2 欄和第 1 欄，以確保此查詢每次執行時，Amazon Redshift 會傳回一致排序的結果。例如，銷售 ID 為 10001 和 10006 的列有相同的 QTY 和 RNK 值。依第 1 欄排序最終結果集可確保列 10001 一定位於 10006 之前。如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

在以下範例中，範圍函數反向排序 (order by qty desc)。現在，最高排名值套用至最大 QTY 值。

```
select salesid, qty,
rank() over (order by qty desc) as rank
from winsales
order by 2,1;

 salesid | qty | rank
---------+-----+-----
   10001 |  10 |   8
   10006 |  10 |   8
   30001 |  10 |   8
   40005 |  10 |   8
   30003 |  15 |   7
   20001 |  20 |   4
   20002 |  20 |   4
   30004 |  20 |   4
   10005 |  30 |   2
   30007 |  30 |   2
   40001 |  40 |   1
(11 rows)
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

下列範例會依 SELLERID 分割資料表，並依數量排序每一個分割區 (以遞減順序)，然後指派排名給每一列。套用範圍函數結果之後排序結果。

```
select salesid, sellerid, qty, rank() over
(partition by sellerid
order by qty desc) as rank
from winsales
order by 2,3,1;

salesid | sellerid | qty | rank
--------+----------+-----+-----
  10001 |        1 |  10 |  2
  10006 |        1 |  10 |  2
  10005 |        1 |  30 |  1
  20001 |        2 |  20 |  1
  20002 |        2 |  20 |  1
  30001 |        3 |  10 |  4
  30003 |        3 |  15 |  3
  30004 |        3 |  20 |  2
```

```
  30007 |        3 |  30 |  1
  40005 |        4 |  10 |  2
  40001 |        4 |  40 |  1
(11 rows)
```

RATIO_TO_REPORT 範圍函數

計算視窗或分割區內值與值總和的比率。使用下列公式決定報告值的比率：

`value of` ratio_expression `argument for the current row / sum of` ratio_expression `argument for the window or partition`

以下資料集示範此公式的使用：

```
Row# Value Calculation RATIO_TO_REPORT
1 2500 (2500)/(13900) 0.1798
2 2600 (2600)/(13900) 0.1870
3 2800 (2800)/(13900) 0.2014
4 2900 (2900)/(13900) 0.2086
5 3100 (3100)/(13900) 0.2230
```

傳回值範圍是 0 至 1 (含)。如果 ratio_expression 為 NULL，則傳回值為 NULL。如果 partition_expression 中的值是唯一的，則函數將傳回該值。

語法

```
RATIO_TO_REPORT ( ratio_expression )
OVER ( [ PARTITION BY partition_expression ] )
```

引數

ratio_expression

此表達式 (例如欄名) 提供要決定比率的值。表達式必須為數值資料類型，或可隱含地轉換為數值資料類型。

您不能在 ratio_expression 中使用其他任何分析函數。

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗排序或視窗框規格。

PARTITION BY partition_expression

選用。此表達式針對 OVER 子句中的每一個群組，設定記錄範圍。

傳回類型

FLOAT8

範例

下列範例使用 WINSALES 資料表。如需有關如何建立 WINSALES 資料表的資訊，請參閱範圍函數範例的範例資料表。

下列範例會計算賣家數量的每一列，以及所有賣家數量的總數。 ratio-to-report

```
select sellerid, qty, ratio_to_report(qty)
over()
from winsales
order by sellerid;

sellerid  qty    ratio_to_report
----------------------------------------
1         30     0.13953488372093023
1         10     0.046511627906976744
1         10     0.046511627906976744
2         20     0.09302325581395349
2         20     0.09302325581395349
3         30     0.13953488372093023
3         20     0.09302325581395349
3         15     0.06976744186046512
3         10     0.046511627906976744
4         10     0.046511627906976744
4         40     0.18604651162790697
```

以下範例依分割區計算每一個賣方的銷售數量比例。

```
select sellerid, qty, ratio_to_report(qty)
over(partition by sellerid)
from winsales;

sellerid   qty    ratio_to_report
-----------------------------------------
2          20     0.5
```

```
2          20     0.5
4          40     0.8
4          10     0.2
1          10     0.2
1          30     0.6
1          10     0.2
3          10     0.13333333333333333
3          15     0.2
3          20     0.26666666666666666
3          30     0.4
```

ROW_NUMBER 範圍函數

根據 OVER 子句中的 ORDER BY 運算式，指派一組列之內目前列的序數 (從 1 起算)。如果有選用的 PARTITION BY 子句，則會重設每一組列的序數。對於 ORDER BY 表達式，具有相等值的列會獲得非決定性的不同列號。

語法

```
ROW_NUMBER() OVER(
  [ PARTITION BY expr_list ]
  [ ORDER BY order_list ]
)
```

引數

( )

此函數不接受引數，但需要空括號。

OVER

ROW_NUMBER 函數的範圍函數子句。

PARTITION BY expr_list

選用。將結果分割成多組列的一或多個欄運算式。

ORDER BY order_list

選用。定義集合中資料列順序的一或多個資料行運算式。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

如果 ORDER BY 未產生唯一排序或被省略，則列的順序不確定。如需詳細資訊，請參閱 範圍函數的資料唯一排序。

傳回類型

BIGINT

範例

下列範例使用 WINSALES 資料表。如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

下列範例依數量對資料表進行排序 (以遞增順序)，然後將列號指派給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, sellerid, qty,
ROW_NUMBER() OVER(
   ORDER BY qty ASC) AS row
FROM winsales
ORDER BY 4,1;

salesid   sellerid   qty   row
---------+----------+-----+-----
   30001 |        3 |  10 |   1
   10001 |        1 |  10 |   2
   10006 |        1 |  10 |   3
   40005 |        4 |  10 |   4
   30003 |        3 |  15 |   5
   20001 |        2 |  20 |   6
   20002 |        2 |  20 |   7
   30004 |        3 |  20 |   8
   10005 |        1 |  30 |   9
   30007 |        3 |  30 |  10
   40001 |        4 |  40 |  11
```

下列範例依 SELLERID 分割資料表，並依 QTY 排序每一個分割區 (以遞增順序)，然後將列號指派給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, sellerid, qty,
ROW_NUMBER() OVER(
  PARTITION BY sellerid
  ORDER BY qty ASC) AS row_by_seller
FROM winsales
ORDER BY 2,4;

 salesid | sellerid | qty | row_by_seller
```

```
---------+----------+-----+-----
   10001 |        1 |  10 |   1
   10006 |        1 |  10 |   2
   10005 |        1 |  30 |   3
   20001 |        2 |  20 |   1
   20002 |        2 |  20 |   2
   30001 |        3 |  10 |   1
   30003 |        3 |  15 |   2
   30004 |        3 |  20 |   3
   30007 |        3 |  30 |   4
   40005 |        4 |  10 |   1
   40001 |        4 |  40 |   2
```

下列範例顯示不使用選用子句時的結果。

```
SELECT salesid, sellerid, qty, ROW_NUMBER() OVER() AS row
FROM winsales
ORDER BY 4,1;

salesid   sellerid   qty   row
---------+----------+-----+-----
   30001 |        3 |  10 |   1
   10001 |        1 |  10 |   2
   10005 |        1 |  30 |   3
   40001 |        4 |  40 |   4
   10006 |        1 |  10 |   5
   20001 |        2 |  20 |   6
   40005 |        4 |  10 |   7
   20002 |        2 |  20 |   8
   30003 |        3 |  15 |   9
   30004 |        3 |  20 |  10
   30007 |        3 |  30 |  11
```

STDDEV_SAMP 和 STDDEV_POP 範圍函數

STDDEV_SAMP 和 STDDEV_POP 範圍函數傳回一組數值 (整數、小數或浮點數) 的樣本標準差和母體標準差。另請參閱STDDEV_SAMP 和 STDDEV_POP 函數。

STDDEV_SAMP 和 STDDEV 是同一個函數的同義詞。

語法

```
STDDEV_SAMP | STDDEV | STDDEV_POP
```

```
( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

如果指定引數 ALL，函數會保留表達式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句區分視窗彙總函數和正常組彙總函數。

PARTITION BY expr_list

以一或多個表達式定義函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

資料類型

STDDEV 函數支援的引數類型包括 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL 及 DOUBLE PRECISION。

不論表達式的資料類型，STDDEV 函數的傳回類型都是雙精確度數字。

範例

下列範例顯示如何使用 STDDEV_POP 和 VAR_POP 函數做為範圍函數。查詢計算 SALES 資料表中 PRICEPAID 值的母體變異數和母體標準差。

```
select salesid, dateid, pricepaid,
round(stddev_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as stddevpop,
round(var_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as varpop
from sales
order by 2,1;

salesid | dateid | pricepaid | stddevpop | varpop
--------+--------+-----------+-----------+---------
  33095 |   1827 |    234.00 |         0 |       0
  65082 |   1827 |    472.00 |       119 |   14161
  88268 |   1827 |    836.00 |       248 |   61283
  97197 |   1827 |    708.00 |       230 |   53019
 110328 |   1827 |    347.00 |       223 |   49845
 110917 |   1827 |    337.00 |       215 |   46159
 150314 |   1827 |    688.00 |       211 |   44414
 157751 |   1827 |   1730.00 |       447 |  199679
 165890 |   1827 |   4192.00 |      1185 | 1403323
...
```

樣本標準差和變異數函數可如法泡製。

SUM 範圍函數

SUM 範圍函數傳回輸入欄或表達式值的總和。SUM 函數處理數值，且忽略 NULL 值。

語法

```
SUM ( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

如果指定引數 ALL，函數會保留表達式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句區分視窗彙總函數和正常組彙總函數。

PARTITION BY expr_list

以一或多個表達式定義 SUM 函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

資料類型

SUM 函數支援的引數類型包括 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL 及 DOUBLE PRECISION。

SUM 函數支援的傳回類型如下：

- BIGINT 代表 SMALLINT 或 INTEGER 引數
- NUMERIC 代表 BIGINT 引數
- DOUBLE PRECISION 代表浮點數引數

範例

下列範例會建立銷售數量的累積 (滾動) 總和，依日期和銷售 ID 排序：

```
select salesid, dateid, sellerid, qty,
sum(qty) over (order by dateid, salesid rows unbounded preceding) as sum
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | sum
---------+------------+----------+-----+-----
```

```
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  20
10005 | 2003-12-24 |        1 |  30 |  50
40001 | 2004-01-09 |        4 |  40 |  90
10006 | 2004-01-18 |        1 |  10 | 100
20001 | 2004-02-12 |        2 |  20 | 120
40005 | 2004-02-12 |        4 |  10 | 130
20002 | 2004-02-16 |        2 |  20 | 150
30003 | 2004-04-18 |        3 |  15 | 165
30004 | 2004-04-18 |        3 |  20 | 185
30007 | 2004-09-07 |        3 |  30 | 215
(11 rows)
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

以下範例會依日期建立銷售數量的累積 (滾動) 總和、依賣方 ID 分割結果，然後在分割區內依日期和銷售 ID 排序結果：

```
select salesid, dateid, sellerid, qty,
sum(qty) over (partition by sellerid
order by dateid, salesid rows unbounded preceding) as sum
from winsales
order by 2,1;

salesid |   dateid   | sellerid | qty | sum
---------+------------+----------+-----+-----
30001 | 2003-08-02 |        3 |  10 |  10
10001 | 2003-12-24 |        1 |  10 |  10
10005 | 2003-12-24 |        1 |  30 |  40
40001 | 2004-01-09 |        4 |  40 |  40
10006 | 2004-01-18 |        1 |  10 |  50
20001 | 2004-02-12 |        2 |  20 |  20
40005 | 2004-02-12 |        4 |  10 |  50
20002 | 2004-02-16 |        2 |  20 |  40
30003 | 2004-04-18 |        3 |  15 |  25
30004 | 2004-04-18 |        3 |  20 |  45
30007 | 2004-09-07 |        3 |  30 |  75
(11 rows)
```

以下範例將結果集的所有列依序編號，依 SELLERID 和 SALESID 欄排序：

```
select salesid, sellerid, qty,
sum(1) over (order by sellerid, salesid rows unbounded preceding) as rownum
```

```
from winsales
order by 2,1;

salesid | sellerid |  qty | rownum
--------+----------+------+--------
10001 |        1 |   10 |      1
10005 |        1 |   30 |      2
10006 |        1 |   10 |      3
20001 |        2 |   20 |      4
20002 |        2 |   20 |      5
30001 |        3 |   10 |      6
30003 |        3 |   15 |      7
30004 |        3 |   20 |      8
30007 |        3 |   30 |      9
40001 |        4 |   40 |     10
40005 |        4 |   10 |     11
(11 rows)
```

如需 WINSALES 資料表的描述，請參閱範圍函數範例的範例資料表。

以下範例將結果集的所有列依序編號、依 SELLERID 分割結果，然後在分割區內依 SELLERID 和 SALESID 排序結果：

```
select salesid, sellerid, qty,
sum(1) over (partition by sellerid
order by sellerid, salesid rows unbounded preceding) as rownum
from winsales
order by 2,1;

salesid | sellerid | qty | rownum
---------+----------+-----+--------
10001 |        1 |  10 |      1
10005 |        1 |  30 |      2
10006 |        1 |  10 |      3
20001 |        2 |  20 |      1
20002 |        2 |  20 |      2
30001 |        3 |  10 |      1
30003 |        3 |  15 |      2
30004 |        3 |  20 |      3
30007 |        3 |  30 |      4
40001 |        4 |  40 |      1
40005 |        4 |  10 |      2
(11 rows)
```

VAR_SAMP 和 VAR_POP 範圍函數

VAR_SAMP 和 VAR_POP 範圍函數傳回一組數值 (整數、小數或浮點數) 的樣本變異數和母體變異數。另請參閱VAR_SAMP 和 VAR_POP 函數。

VAR_SAMP 和 VARIANCE 是同一個函數的同義詞。

語法

```
VAR_SAMP | VARIANCE | VAR_POP
( [ ALL ] expression ) OVER
(
[ PARTITION BY expr_list ]
[ ORDER BY order_list
                        frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

如果指定引數 ALL，函數會保留表達式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句區分視窗彙總函數和正常組彙總函數。

PARTITION BY expr_list

以一或多個表達式定義函數的視窗。

ORDER BY order_list

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDER BY 會使用整個資料表。

frame_clause

如果彙總函數使用 ORDER BY 子句，則需要明確的窗框子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。窗框子句包含 ROWS 關鍵字和相關的指定元。請參閱範圍函數語法摘要。

資料類型

VARIANCE 函數支援的引數類型包括
SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL 及 DOUBLE PRECISION。

不論表達式的資料類型，VARIANCE 函數的傳回類型都是雙精確度數字。

系統管理函數

主題

Amazon Redshift 支援多個系統管理函數。

CHANGE_QUERY_PRIORITY

CHANGE_QUERY_PRIORITY 可讓超級使用者修改在工作負載管理 (WLM) 中執行或等待之查詢的優先順序。

此函數讓超級使用者可以立即變更系統中任何查詢的優先順序。只有一個查詢、使用者或工作階段可以使用優先順序 CRITICAL 執行。

語法

```
CHANGE_QUERY_PRIORITY(query_id, priority)
```

引數

query_id

要變更其優先順序之查詢的查詢識別碼。需要 INTEGER 值。

priority

要指派給查詢的新優先順序。這個引數必須是具備 CRITICAL、HIGHEST、HIGH、NORMAL、LOW 或 LOWEST 值的字串。

傳回類型

無

範例

若要顯示 STV_WLM_QUERY_STATE 系統資料表中的 query_priority 欄，請使用下列範例。

```
SELECT query, service_class, query_priority, state
FROM stv_wlm_query_state WHERE service_class = 101;

+-------+---------------+----------------+---------+
| query | service_class | query_priority |  state  |
+-------+---------------+----------------+---------+
|  1076 |           101 | Lowest         | Running |
|  1075 |           101 | Lowest         | Running |
+-------+---------------+----------------+---------+
```

若要顯示超級使用者執行函數 change_query_priority 以將優先順序變更為 CRITICAL 的結果，請使用下列範例。

```
SELECT CHANGE_QUERY_PRIORITY(1076, 'Critical');

+-----------------------------------------------------------------------------------+
|                             change_query_priority                                 |
+-----------------------------------------------------------------------------------+
| Succeeded to change query priority. Priority changed from Lowest to Critical. |
+-----------------------------------------------------------------------------------+
```

CHANGE_SESSION_PRIORITY

CHANGE_SESSION_PRIORITY 讓超級使用者可以立即變更系統中任何工作階段的優先順序。只有一個工作階段、使用者或查詢可以使用優先順序 CRITICAL 執行。

語法

```
CHANGE_SESSION_PRIORITY(pid, priority)
```

引數

pid

要變更其優先順序之工作階段的程序識別碼。值 -1 表示目前工作階段。需要 INTEGER 值。

priority

要指派給工作階段的新優先順序。這個引數必須是具備 CRITICAL、HIGHEST、HIGH、NORMAL、LOW 或 LOWEST 值的字串。

傳回類型

無

範例

若要傳回處理目前工作階段之伺服器處理程序的程序識別碼，請使用下列範例。

```
SELECT pg_backend_pid();

+-----------------+
| pg_backend_pid |
+-----------------+
|          30311 |
+-----------------+
```

在此範例中，將目前工作階段的優先順序變更為 LOWEST。

```
SELECT CHANGE_SESSION_PRIORITY(30311, 'Lowest');

+-------------------------------------------------------------------------------------
+
|                                change_session_priority
 |
+-------------------------------------------------------------------------------------
+
```

```
| Succeeded to change session priority. Changed session (pid:30311) priority to lowest.
 |
+-------------------------------------------------------------------------------------
+
```

在此範例中，將目前工作階段的優先順序變更為 HIGH。

```
SELECT CHANGE_SESSION_PRIORITY(-1, 'High');

+-------------------------------------------------------------------------------------------------------
+
|                                    change_session_priority
           |
+-------------------------------------------------------------------------------------------------------
+
| Succeeded to change session priority. Changed session (pid:30311) priority from
 lowest to high. |
+-------------------------------------------------------------------------------------------------------
+
```

若要建立變更工作階段優先順序的預存程序，請使用下列範例。將執行此預存程序的許可授予給資料庫使用者test_user。

```
CREATE OR REPLACE PROCEDURE sp_priority_low(pid IN int, result OUT varchar)
AS $$
BEGIN
  SELECT CHANGE_SESSION_PRIORITY(pid, 'low') into result;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER;
GRANT EXECUTE ON PROCEDURE sp_priority_low(int) TO test_user;
```

接著名為 test_user 的資料庫使用者呼叫程序。

```
CALL sp_priority_low(pg_backend_pid());

+-----------------------------------------------------+
|                       result                        |
+-----------------------------------------------------+
| Success. Change session (pid:13155) priority to low. |
+-----------------------------------------------------+
```

CHANGE_USER_PRIORITY

CHANGE_USER_PRIORITY 可讓超級使用者修改在工作負載管理 (WLM) 中，由某位使用者發出之所有執行或等待中查詢的優先順序。只有一個使用者、工作階段或查詢可以使用優先順序 CRITICAL 執行。

語法

```
CHANGE_USER_PRIORITY(user_name, priority)
```

引數

user_name

要變更其查詢優先順序的資料庫使用者名稱。

priority

要指派給由 user_name 發出之所有查詢的新優先順序。這個引數必須是具備 CRITICAL、HIGHEST、HIGH、NORMAL、LOW、LOWEST 或 RESET 值的字串。只有超級使用者才能將優先順序變更為 CRITICAL。將優先順序變更為 RESET，會移除 user_name 的優先順序設定。

傳回類型

無

範例

若要將使用者 analysis_user 的優先順序變更為 LOWEST，請使用下列範例。

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'lowest');

+-------------------------------------------------------------------------------------+
|                                change_user_priority                                 |
+-------------------------------------------------------------------------------------+
| Succeeded to change user priority. Changed user (analysis_user) priority to lowest. |
+-------------------------------------------------------------------------------------+
```

若要將優先順序變更為 LOW，請使用下列範例。

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'low');
```

```
+-----------------------------------------------------------------------------------------------
+
|                                    change_user_priority
        |
+-----------------------------------------------------------------------------------------------
+
| Succeeded to change user priority. Changed user (analysis_user) priority from Lowest
 to low. |
+-----------------------------------------------------------------------------------------------
+
```

若要重設優先順序，請使用下列範例。

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'reset');

+-----------------------------------------------------+
|                 change_user_priority                |
+-----------------------------------------------------+
| Succeeded to reset priority for user (analysis_user). |
+-----------------------------------------------------+
```

CURRENT_SETTING

CURRENT_SETTING 傳回指定之組態參數的目前值。

此函數相當於 SHOW 命令。

語法

```
current_setting('parameter')
```

以下陳述式會傳回指定工作階段內容變數的目前值。

```
current_setting('variable_name')
current_setting('variable_name'[, error_if_undefined])
```

引數

parameter

要顯示的參數值。如需組態參數的清單，請參閱組態參考

variable_name

要顯示的變數名稱。這必須是工作階段內容變數的字串常數。

error_if_undefined

(選用) Boolean 值，指定變數名稱不存在時的行為。當 error_if_undefined 設定為 TRUE (這是預設設定)，Amazon Redshift 擲回錯誤。當 error_if_undefined 設定為 FALSE，Amazon Redshift 傳回 NULL。Amazon Redshift 僅支援工作階段內容變數的 error_if_undefined 參數。當輸入是組態參數時，不能使用此選項。

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

若要傳回 query_group 參數的目前設定，請使用下列範例。

```
SELECT CURRENT_SETTING('query_group');

+-----------------+
| current_setting |
+-----------------+
| unset           |
+-----------------+
```

若要傳回變數 app_context.user_id 的目前設定，請使用下列範例。

```
SELECT CURRENT_SETTING('app_context.user_id', FALSE);
```

PG_CANCEL_BACKEND

取消查詢。PG_CANCEL_BACKEND 的功能相當於 取消 命令。您可以取消使用者目前正在執行的查詢。超級使用者可以取消任何查詢。

語法

```
pg_cancel_backend( pid )
```

引數

pid

要取消之查詢的處理程序 ID (PID)。您無法藉由指定查詢 ID 來取消查詢；您必須指定查詢的處理程序 ID。需要 INTEGER 值。

傳回類型

無

使用須知

如果有多個工作階段中的查詢在同一資料表上保持鎖定，您可以使用 PG_TERMINATE_BACKEND 函數來終止其中一個工作階段，這樣可強制終止工作階段中任何目前正在執行的交易，以解除所有鎖定並恢復交易。查詢 PG__LOCKS 目錄資料表以檢視目前持有的鎖定。如果因為查詢在交易區塊中 (BEGIN … END) 而無法取消查詢，您可以使用 PG_TERMINATE_BACKEND 函數，以終止查詢執行所在的工作階段。

範例

若要取消目前執行中的查詢，請先擷取您要取消之查詢的處理程序 ID。若要判斷所有目前執行中查詢的處理程序 ID，請執行下列命令。

```
SELECT pid, TRIM(starttime) AS start,
duration, TRIM(user_name) AS user,
SUBSTRING(query,1,40) AS querytxt
FROM stv_recents
WHERE status = 'Running';

+-----+-------------------------+----------+--------+-----------------------------+
| pid |       starttime         | duration |  user  |          querytxt           |
+-----+-------------------------+----------+--------+-----------------------------+
| 802 | 2013-10-14 09:19:03.55 |      132 | dwuser | select venuename from venue |
| 834 | 2013-10-14 08:33:49.47 |  1250414 | dwuser | select * from listing;      |
| 964 | 2013-10-14 08:30:43.29 |   326179 | dwuser | select sellerid from sales  |
+-----+-------------------------+----------+--------+-----------------------------+
```

若要取消處理程序識別碼為 802 的查詢，請使用下列範例。

```
SELECT PG_CANCEL_BACKEND(802);
```

PG_TERMINATE_BACKEND

終止工作階段。您可以終止使用者所擁有的工作階段。超級使用者可以終止任何工作階段。

語法

```
pg_terminate_backend( pid )
```

引數

pid

要終止之工作階段的處理程序 ID。需要 INTEGER 值。

傳回類型

無

使用須知

如果快要接近並行連線數上限，請使用 PG_TERMINATE_BACKEND 終止閒置工作階段並釋出連線。如需詳細資訊，請參閱 Amazon Redshift 限制。

如果有多個工作階段中的查詢在同一資料表上保持鎖定，您可以使用 PG_TERMINATE_BACKEND 來終止其中一個工作階段，這樣可強制終止工作階段中任何目前正在執行的交易，以解除所有鎖定並恢復交易。查詢 PG__LOCKS 目錄資料表以檢視目前持有的鎖定。

如果查詢不在交易區塊中 (BEGIN … END)，您可以使用 取消 命令或 PG_CANCEL_BACKEND 函數來取消查詢。

範例

若要查詢 SVV_TRANSACTIONS 資料表來檢視目前交易中生效的所有鎖定，請使用下列範例。

```
SELECT * FROM svv_transactions;

+-----------+--------+-------+------+--------------------+-----------------
+----------------------+----------+---------+
| txn_owner | txn_db |  xid  | pid  |      txn_start      |    lock_mode    |
 lockable_object_type | relation | granted |
+-----------+--------+-------+------+--------------------+-----------------
+----------------------+----------+---------+
```

```
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
            |    51940 | true    |
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
            |    52000 | true    |
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
            |   108623 | true    |
| rsuser    | dev    | 96178 | 8585 | 2017-04-12 20:13:07 | ExclusiveLock   |
 transactionid        |          | true    |
+-----------+--------+-------+------+---------------------+-----------------
+----------------------+----------+---------+
```

若要終止保持鎖定的工作階段，請使用下列範例。

```
SELECT PG_TERMINATE_BACKEND(8585);
```

REBOOT_CLUSTER

重新啟動 Amazon Redshift 叢集，而不關閉與叢集的連線。您必須是資料庫超級使用者才能執行此命令。

完成此軟重新開機之後，Amazon Redshift 叢集會將錯誤傳回給使用者應用程式，並要求使用者應用程式重新提交任何因軟重新開機而中斷的交易或查詢。

語法

```
SELECT REBOOT_CLUSTER();
```

SET_CONFIG

將組態參數設為新的設定。

此函數相當於 SQL 中的 SET 命令。

語法

```
SET_CONFIG('parameter', 'new_value' , is_local)
```

下列陳述式會將工作階段內容變數設定為新設定。

```
set_config('variable_name', 'new_value' , is_local)
```

引數

parameter

要設定的參數。

variable_name

要設定的變數名稱。

new_value

參數的新值。

is_local

若為 true，參數值僅套用至目前交易。有效值為 `true` 或 `1` 及 `false` 或 `0`。

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

若要僅針對目前交易將 `query_group` 參數的值設為 `test`，請使用下列範例。

```
SELECT SET_CONFIG('query_group', 'test', true);

+------------+
| set_config |
+------------+
| test       |
+------------+
```

若要設定工作階段內容變數，請使用下列範例。

```
SELECT SET_CONFIG(‘app.username’, ‘cuddy’, FALSE);
```

系統資訊函數

Amazon Redshift 支援許多系統資訊函數。

主題

- CURRENT_AWS_ACCOUNT
- CURRENT_DATABASE
- CURRENT_NAMESPACE
- CURRENT_SCHEMA
- CURRENT_SCHEMAS
- CURRENT_USER
- CURRENT_USER_ID
- DEFAULT_IAM_ROLE
- HAS_ASSUMEROLE_PRIVILEGE
- HAS_DATABASE_PRIVILEGE
- HAS_SCHEMA_PRIVILEGE
- HAS_TABLE_PRIVILEGE
- LAST_USER_QUERY_ID
- PG_BACKEND_PID
- PG_GET_COLS
- PG_GET_GRANTEE_BY_IAM_ROLE
- PG_GET_IAM_ROLE_BY_USER
- PG_GET_LATE_BINDING_VIEW_COLS
- PG_GET_SESSION_ROLES
- PG_LAST_COPY_COUNT
- PG_LAST_COPY_ID
- PG_LAST_UNLOAD_ID
- PG_LAST_QUERY_ID
- PG_LAST_UNLOAD_COUNT
- SLICE_NUM 函數
- USER
- ROLE_IS_MEMBER_OF
- USER_IS_MEMBER_OF
- VERSION

CURRENT_AWS_ACCOUNT

傳回與提交查詢之 Amazon Redshift 叢集相關聯的 AWS 帳戶。

語法

```
current_aws_account
```

傳回類型

傳回整數。

範例

下列查詢傳回目前資料庫的名稱。

```
select user, current_aws_account;
current_user | current_account
-------------+---------------
dwuser       | 987654321

(1 row)
```

CURRENT_DATABASE

傳回您目前連接的資料庫名稱。

語法

```
current_database()
```

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列查詢傳回目前資料庫的名稱。

```
select current_database();

current_database
------------------
```

```
tickit
(1 row)
```

CURRENT_NAMESPACE

傳回目前 Amazon Redshift 叢集的叢集命名空間。Amazon Redshift 叢集命名空間是 Amazon Redshift 叢集的唯一 ID。

語法

```
current_namespace
```

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列查詢傳回目前命名空間的名稱。

```
select user, current_namespace;
current_user | current_namespace
-------------+--------------------------------------
dwuser       | 86b5169f-01dc-4a6f-9fbb-e2e24359e9a8

(1 row)
```

CURRENT_SCHEMA

傳回搜尋路徑前面的結構描述名稱。此結構描述將用於建立時未指定目標結構描述的任何資料表或其他具名物件。

語法

> **Note**
>
> 這是領導者節點函數。此函數在參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視時會傳回錯誤。

```
current_schema()
```

傳回類型

CURRENT_SCHEMA 傳回 CHAR 或 VARCHAR 字串。

範例

下列查詢傳回目前結構描述：

```
select current_schema();

current_schema
---------------
public
(1 row)
```

CURRENT_SCHEMAS

傳回目前搜尋路徑中任何結構描述的名稱所構成的陣列。目前搜尋路徑是在 search_path 參數中定義。

語法

> **Note**
>
> 這是領導者節點函數。此函數在參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視時會傳回錯誤。

```
current_schemas(include_implicit)
```

引數

include_implicit

若為 true，表示搜尋路徑應該包含任何隱含包含的系統結構描述。有效值為 `true` 和 `false`。通常，若為 `true`，此參數除了傳回目前的結構描述外，還會傳回 `pg_catalog` 結構描述。

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列範例傳回目前搜尋路徑中的結構描述名稱，但不包括隱含包含的系統結構描述：

```
select current_schemas(false);

current_schemas
-----------------
{public}
(1 row)
```

下列範例傳回目前搜尋路徑中的結構描述名稱，包括隱含包含的系統結構描述：

```
select current_schemas(true);

current_schemas
---------------------
{pg_catalog,public}
(1 row)
```

CURRENT_USER

傳回資料庫的目前「有效」使用者的使用者名稱，適用於檢查權限。通常，此使用者名稱與工作階段使用者相同；不過，超級使用者偶爾會改變這種情況。

> **Note**
>
> 呼叫 CURRENT_USER 時，請勿使用結尾括號。

語法

```
current_user
```

傳回類型

CURRENT_USER 會傳回名稱資料類型，並且可以轉換為字元或 VARCHAR 字串。

使用須知

如果預存程序是使用 CREATE_PROCURCE 命令的 SECURITY DEFINER 選項建立的，則當從預存程序中調用 CURRENT_USER 函數時，Amazon Redshift 會傳回預存程序擁有者的使用者名稱。

範例

下列查詢傳回目前資料庫使用者的名稱：

```
select current_user;

current_user
--------------
dwuser
(1 row)
```

CURRENT_USER_ID

傳回登入目前工作階段之 Amazon Redshift 使用者的唯一識別碼。

語法

```
CURRENT_USER_ID
```

傳回類型

CURRENT_USER_ID 函數傳回整數。

範例

下列範例傳回此工作階段的使用者名稱和目前使用者 ID：

```
select user, current_user_id;

 current_user | current_user_id
--------------+-----------------
   dwuser     |               1
(1 row)
```

DEFAULT_IAM_ROLE

傳回目前與 Amazon Redshift 叢集關聯的預設 IAM 角色。如果沒有任何關聯的預設 IAM 角色，則函數將不傳回任何內容。

語法

```
select default_iam_role();
```

傳回類型

傳回 VARCHAR 字串。

範例

下列範例會傳回目前與指定的 Amazon Redshift 叢集關聯的預設 IAM 角色，

```
select default_iam_role();
              default_iam_role
---------------------------------------------
 arn:aws:iam::123456789012:role/myRedshiftRole
(1 row)
```

HAS_ASSUMEROLE_PRIVILEGE

如果指定的使用者具有具有執行指定命令之權限的指定 IAM 角色，則傳回布林值 `true` (t)。如果使用者不具有執行指定命令之權限的指定 IAM 角色，則函數會傳回 `false` (f)。如需權限的相關資訊，請參閱 GRANT。

語法

```
has_assumerole_privilege( [ user, ] iam_role_arn, cmd_type)
```

引數

使用者

要檢查 IAM 角色權限的使用者名稱。預設值是檢查目前使用者。超級使用者和使用者可以使用此函數。不過，使用者只能檢視自己的權限。

iam_role_arn

已被授與命令權限的 IAM 角色。

cmd_type

已授與存取權的命令。有效值如下：

- COPY
- UNLOAD
- 外部函數
- 建立模型

傳回類型

BOOLEAN

範例

下列查詢會確認使用者 reg_user1 具有執行 COPY 命令之 Redshift-S3-Read 角色的權限。

```
select has_assumerole_privilege('reg_user1', 'arn:aws:iam::123456789012:role/Redshift-S3-Read', 'copy');
```

```
has_assumerole_privilege
------------------------
true
(1 row)
```

HAS_DATABASE_PRIVILEGE

如果使用者具有指定之資料庫的指定權限，則傳回 true。如需權限的相關資訊，請參閱 GRANT。

語法

> **Note**
>
> 這是領導者節點函數。此函數在參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視時會傳回錯誤。

```
has_database_privilege( [ user, ] database, privilege)
```

引數

使用者

要檢查資料庫權限的使用者名稱。預設值是檢查目前使用者。

資料庫

與權限相關聯的資料庫。

privilege

要檢查的權限。有效值如下：

- CREATE
- TEMPORARY
- TEMP

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列查詢確認 GUEST 使用者在 TICKIT 資料庫上具有 TEMP 權限：

```
select has_database_privilege('guest', 'tickit', 'temp');

has_database_privilege
-----------------------
true
(1 row)
```

HAS_SCHEMA_PRIVILEGE

如果使用者具有指定之結構描述的指定權限，則傳回 `true`。如需權限的相關資訊，請參閱 GRANT。

語法

> **Note**
>
> 這是領導者節點函數。此函數在參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視時會傳回錯誤。

```
has_schema_privilege( [ user, ] schema, privilege)
```

引數

使用者

要檢查結構描述權限的使用者名稱。預設值是檢查目前使用者。

結構描述

與權限相關聯的結構描述。

privilege

要檢查的權限。有效值如下：

- CREATE
- USAGE

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列查詢確認 GUEST 使用者在 PUBLIC 結構描述上具有 CREATE 權限：

```
select has_schema_privilege('guest', 'public', 'create');

has_schema_privilege
----------------------
true
(1 row)
```

HAS_TABLE_PRIVILEGE

如果使用者具有指定之資料表的指定權限，則傳回 `true`，否則傳回 `false`。

語法

> **Note**
>
> 這是領導者節點函數。此函數在參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視時會傳回錯誤。如需權限的相關資訊，請參閱 GRANT。

```
has_table_privilege( [ user, ] table, privilege)
```

引數

使用者

要檢查資料表權限的使用者名稱。預設值是檢查目前使用者。

表格

與權限相關聯的資料表。

privilege

要檢查的權限。有效值如下：

- SELECT
- INSERT
- UPDATE
- DELETE
- DROP
- REFERENCES

傳回類型

BOOLEAN

範例

下列查詢發現 GUEST 使用者在 LISTING 資料表上沒有 SELECT 權限。

```
select has_table_privilege('guest', 'listing', 'select');

has_table_privilege
---------------------
false
```

下列查詢會使用 pg_tables 和 pg_user 目錄資料表的輸出，列出資料表權限，包括選取、插入、更新和刪除。這只是範例。您可能必須從資料庫中指定結構描述名稱和資料表名稱。如需詳細資訊，請參閱查詢目錄資料表。

```
SELECT
     tablename
     ,usename
```

```
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'select') AS sel
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'insert') AS ins
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'update') AS upd
     ,HAS_TABLE_PRIVILEGE(users.usename, tablename, 'delete') AS del
FROM
(SELECT * from pg_tables
WHERE schemaname = 'public' and tablename in ('event','listing')) as tables
,(SELECT * FROM pg_user) AS users;

tablename | usename   |  sel   |  ins   |  upd   | del
----------+-----------+--------+--------+--------+-------
event     |  john     |  true  | true   | true   | true
event     |  sally    |  false | false  | false  | false
event     |  elsa     |  false | false  | false  | false
listing   |  john     |  true  | true   | true   | true
listing   |  sally    |  false | false  | false  | false
listing   |  elsa     |  false | false  | false  | false
```

先前的查詢也包含交叉聯結。如需詳細資訊，請參閱 JOIN 範例。若要查詢不在 public 結構描述中的資料表，請從 WHERE 子句中移除 schemaname 條件，並在查詢之前使用下列範例。

```
SET SEARCH_PATH to 'schema_name';
```

LAST_USER_QUERY_ID

傳回目前工作階段中最近完成的使用者查詢的查詢 ID。如果目前工作階段中未執行任何查詢，last_user_query_id 會傳回 -1。此函數不會傳回僅在引線節點上執行之查詢的查詢 ID。如需詳細資訊，請參閱 僅限領導節點函數。

語法

```
last_user_query_id()
```

傳回類型

傳回整數。

範例

下列查詢傳回在目前工作階段中完成的使用者最後一個執行查詢的 ID。

```
select last_user_query_id();
```

結果如下。

```
last_user_query_id
-----------------------
    5437
(1 row)
```

下列查詢傳回使用者在目前工作階段中執行的最近完成之查詢的查詢 ID 和文字。

```
select query_id, query_text from sys_query_history where query_id =
 last_user_query_id();
```

結果如下。

```
 query_id, query_text
---------
+-----------------------------------------------------------------------------------------------------------------
 5556975 | select last_user_query_id() limit 100 --RequestID=<unique request ID>;
 TraceID=<unique trace ID>
```

PG_BACKEND_PID

傳回處理目前工作階段之伺服器處理程序的處理程序 ID (PID)。

> **Note**
>
> PID 不是全域唯一。可隨著時間而重複使用。

語法

```
pg_backend_pid()
```

傳回類型

傳回整數。

範例

您可以使 PG_BACKEND_PID 與日誌資料表相互關聯，以擷取目前工作階段的資訊。例如，下列查詢會針對目前工作階段中完成的查詢傳回其查詢 ID 和一部分的查詢文字。

```
select query, substring(text,1,40)
from stl_querytext
where pid =  PG_BACKEND_PID()
order by query desc;

 query |                substring
-------+----------------------------------------
 14831 | select query, substring(text,1,40) from
 14827 | select query, substring(path,0,80) as pa
 14826 | copy category from 's3://dw-tickit/manif
 14825 | Count rows in target table
 14824 | unload ('select * from category') to 's3
(5 rows)
```

您可以使 PG_BACKEND_PID 與下列日誌資料表中的 pid 欄相互關聯 (例外以括號註明)：

- STL_CONNECTION_LOG
- STL_DDLTEXT
- STL_ERROR
- STL_QUERY
- STL_QUERYTEXT
- STL_SESSIONS (process)
- STL_TR_CONFLICT
- STL_UTILITYTEXT
- STV_ACTIVE_CURSORS
- STV_INFLIGHT
- STV_LOCKS (lock_owner_pid)
- STV_RECENTS (process_id)

PG_GET_COLS

傳回資料表或檢視定義的欄中繼資料。

語法

```
pg_get_cols('name')
```

引數

name

Amazon Redshift 資料表或檢視的名稱。如需詳細資訊，請參閱 名稱與識別碼。

傳回類型

VARCHAR

使用須知

PG_GET_COLS 函數針對資料表或檢視定義中的每一欄，各傳回一列。列包含結構描述名稱、關係名稱、欄名稱、資料類型及欄號的逗號分隔清單。SQL 結果的格式取決於所使用的 SQL 用戶端。

範例

下列範例會傳回使用者在連線資料庫SALES_VW中建立的結構描述public和資料表sales中名稱mytickit1的結果，以結構描述命名的檢視表和資料表dev。

下列範例會傳回名為之檢視的資料行中繼資料SALES_VW。

```
select pg_get_cols('sales_vw');

pg_get_cols
---------------------------------------------------------------
(public,sales_vw,salesid,integer,1)
(public,sales_vw,listid,integer,2)
(public,sales_vw,sellerid,integer,3)
(public,sales_vw,buyerid,integer,4)
(public,sales_vw,eventid,integer,5)
(public,sales_vw,dateid,smallint,6)
(public,sales_vw,qtysold,smallint,7)
(public,sales_vw,pricepaid,"numeric(8,2)",8)
(public,sales_vw,commission,"numeric(8,2)",9)
(public,sales_vw,saletime,"timestamp without time zone",10)
```

下列範例會以資料表格式傳回SALES_VW檢視的資料行中繼資料。

```
select * from pg_get_cols('sales_vw')
cols(view_schema name, view_name name, col_name name, col_type varchar, col_num int);

view_schema | view_name | col_name   | col_type                    | col_num
```

```
------------+-----------+------------+------------------------------+--------
public      | sales_vw  | salesid    | integer                      |       1
public      | sales_vw  | listid     | integer                      |       2
public      | sales_vw  | sellerid   | integer                      |       3
public      | sales_vw  | buyerid    | integer                      |       4
public      | sales_vw  | eventid    | integer                      |       5
public      | sales_vw  | dateid     | smallint                     |       6
public      | sales_vw  | qtysold    | smallint                     |       7
public      | sales_vw  | pricepaid  | numeric(8,2)                 |       8
public      | sales_vw  | commission | numeric(8,2)                 |       9
public      | sales_vw  | saletime   | timestamp without time zone  |      10
```

下列範例會以資料表格式傳回結構定義中資料SALES表的資料行mytickit1中繼資料。

```
select * from pg_get_cols('"mytickit1"."sales"')
cols(view_schema name, view_name name, col_name name, col_type varchar, col_num int);

view_schema | view_name | col_name   | col_type                     | col_num
------------+-----------+------------+------------------------------+--------
mytickit1   | sales     | salesid    | integer                      |       1
mytickit1   | sales     | listid     | integer                      |       2
mytickit1   | sales     | sellerid   | integer                      |       3
mytickit1   | sales     | buyerid    | integer                      |       4
mytickit1   | sales     | eventid    | integer                      |       5
mytickit1   | sales     | dateid     | smallint                     |       6
mytickit1   | sales     | qtysold    | smallint                     |       7
mytickit1   | sales     | pricepaid  | numeric(8,2)                 |       8
mytickit1   | sales     | commission | numeric(8,2)                 |       9
mytickit1   | sales     | saletime   | timestamp without time zone  |      10
```

PG_GET_GRANTEE_BY_IAM_ROLE

傳回授與指定 IAM 角色的所有使用者和群組。

語法

```
pg_get_grantee_by_iam_role('iam_role_arn')
```

引數

iam_role_arn

要傳回被授與此角色之使用者和群組的 IAM 角色。

傳回類型

VARCHAR

使用須知

PG_GET_GRANTEE_BY_IAM_ROLE 函數為每個使用者或群組傳回一列。每一列都包含承授者名稱、承授者類型和授與的權限。承授者類型的可能值為 p (代表公用)、u (代表使用者) 和 g (代表群組)。

您必須是超級使用者才能使用此函數。

範例

下列範例指出 IAM 角色 Redshift-S3-Write 已授與 group1 和 reg_user1。group_1 中的使用者只能指定 COPY 作業的角色，而使用者 reg_user1 只能指定角色來執行 UNLOAD 作業。

```
select pg_get_grantee_by_iam_role('arn:aws:iam::123456789012:role/Redshift-S3-Write');
```

```
  pg_get_grantee_by_iam_role
-----------------------------
 (group_1,g,COPY)
 (reg_user1,u,UNLOAD)
```

下列 PG_GET_GRANTEE_BY_IAM_ROLE 函數範例將結果格式化為表格。

```
select grantee, grantee_type, cmd_type FROM
 pg_get_grantee_by_iam_role('arn:aws:iam::123456789012:role/Redshift-S3-Write')
 res_grantee(grantee text, grantee_type text, cmd_type text) ORDER BY 1,2,3;
```

```
  grantee  | grantee_type | cmd_type
-----------+--------------+----------
 group_1   | g            | COPY
 reg_user1 | u            | UNLOAD
```

PG_GET_IAM_ROLE_BY_USER

傳回授與使用者的所有 IAM 角色和命令權限。

語法

```
pg_get_iam_role_by_user('name')
```

引數

name

要傳回 IAM 角色的使用者名稱。

傳回類型

VARCHAR

使用須知

PG_GET_IAM_ROLE_BY_USER 函數會針對每組角色和命令權限傳回一行。此資料列包含以逗號分隔清單，其中包含使用者名稱、IAM 角色和命令。

結果中的 default 值表示使用者可以指定任何可用的角色來執行顯示的命令。

您必須是超級使用者才能使用此函數。

範例

下列範例指出使用者 reg_user1 可以指定任何可用的 IAM 角色來執行 COPY 作業。使用者也可以指定 UNLOAD 作業的 Redshift-S3-Write 角色。

```
select pg_get_iam_role_by_user('reg_user1');
```

```
                            pg_get_iam_role_by_user
-------------------------------------------------------------------------------
 (reg_user1,default,COPY)
 (reg_user1,arn:aws:iam::123456789012:role/Redshift-S3-Write,COPY|UNLOAD)
```

下列 PG_GET_IAM_ROLE_BY_USER 函數範例將結果格式化為表格。

```
select username, iam_role, cmd FROM pg_get_iam_role_by_user('reg_user1')
 res_iam_role(username text, iam_role text, cmd text);
```

```
 username  |                    iam_role                     | cmd
-----------+-------------------------------------------------+------
 reg_user1 | default                                         | None
```

```
 reg_user1 | arn:aws:iam::123456789012:role/Redshift-S3-Read | COPY
```

PG_GET_LATE_BINDING_VIEW_COLS

傳回資料庫中所有近期繫結檢視的欄中繼資料。如需更多資訊，請參閱近期繫結檢視

語法

```
pg_get_late_binding_view_cols()
```

傳回類型

VARCHAR

使用須知

PG_GET_LATE_BINDING_VIEW_COLS 函數針對近期繫結檢視中的每一欄，各傳回一列。列包含結構描述名稱、關係名稱、欄名稱、資料類型及欄號的逗號分隔清單。

範例

下列範例傳回所有近期繫結檢視的欄中繼資料。

```
select pg_get_late_binding_view_cols();

pg_get_late_binding_view_cols
------------------------------------------------------------
(public,myevent,eventname,"character varying(200)",1)
(public,sales_lbv,salesid,integer,1)
(public,sales_lbv,listid,integer,2)
(public,sales_lbv,sellerid,integer,3)
(public,sales_lbv,buyerid,integer,4)
(public,sales_lbv,eventid,integer,5)
(public,sales_lbv,dateid,smallint,6)
(public,sales_lbv,qtysold,smallint,7)
(public,sales_lbv,pricepaid,"numeric(8,2)",8)
(public,sales_lbv,commission,"numeric(8,2)",9)
(public,sales_lbv,saletime,"timestamp without time zone",10)
(public,event_lbv,eventid,integer,1)
(public,event_lbv,venueid,smallint,2)
(public,event_lbv,catid,smallint,3)
(public,event_lbv,dateid,smallint,4)
(public,event_lbv,eventname,"character varying(200)",5)
```

```
(public,event_lbv,starttime,"timestamp without time zone",6)
```

下列範例以資料表格式傳回所有近期繫結檢視的欄中繼資料。

```
select * from pg_get_late_binding_view_cols() cols(view_schema name, view_name name,
 col_name name, col_type varchar, col_num int);
view_schema | view_name | col_name   | col_type                    | col_num
------------+-----------+------------+-----------------------------+--------
public      | sales_lbv | salesid    | integer                     |       1
public      | sales_lbv | listid     | integer                     |       2
public      | sales_lbv | sellerid   | integer                     |       3
public      | sales_lbv | buyerid    | integer                     |       4
public      | sales_lbv | eventid    | integer                     |       5
public      | sales_lbv | dateid     | smallint                    |       6
public      | sales_lbv | qtysold    | smallint                    |       7
public      | sales_lbv | pricepaid  | numeric(8,2)                |       8
public      | sales_lbv | commission | numeric(8,2)                |       9
public      | sales_lbv | saletime   | timestamp without time zone |      10
public      | event_lbv | eventid    | integer                     |       1
public      | event_lbv | venueid    | smallint                    |       2
public      | event_lbv | catid      | smallint                    |       3
public      | event_lbv | dateid     | smallint                    |       4
public      | event_lbv | eventname  | character varying(200)      |       5
public      | event_lbv | starttime  | timestamp without time zone |       6
```

PG_GET_SESSION_ROLES

傳回目前登入使用者的工作階段角色。使用者的工作階段角色是由身分提供者 (IdP) 為登入使用者定義的群組。例如，Microsoft Azure Active Directory (Azure AD) 等身分提供者 (IdP) 會驗證使用者的身分，並在使用者登入程序期間提供使用者所屬的任何外部群組。這些外部群組會轉換為 Amazon Redshift 角色，並可在目前的工作階段期間使用。這些角色稱為工作階段角色。管理員可以授與工作階段角色類似其他 Amazon Redshift 角色的權限。如需使用角色的詳細資訊，請參閱角色型存取控制 (RBAC)。如需使用身分提供者 (IdP) 管理身分的相關資訊，請參閱《Amazon Redshift 管理指南》中的適用於 Amazon Redshift 的原生身分提供者 (IdP) 聯合。

若要檢視 Amazon Redshift 目錄中定義的角色，請以管理員或超級使用者身分連線到資料庫，然後查詢系統檢視 SVV_ROLES。

語法

```
pg_get_session_roles()
```

傳回類型

由兩個值組成的一組列。第一個值有兩個部分，由冒號 (:) 分隔，其中包含 idp-namespace:role-name。idp-namespace 是身分提供者 (IdP) 的命名空間。role-name 是身分提供者 (IdP) 中外部群組的名稱。第二個值包含 role-id，這是角色識別碼。

使用須知

PG_GET_SESSION_ROLES 函數會針對每個傳回的工作階段角色傳回一列。

範例

下列範例會針對 Azure Active Directory IdP 中的每個角色傳回一個資料列。傳回的資料欄會轉換為 sess_roles，具有欄 name 和 roleid。每個 name 都包含 Azure Active Directory 命名空間和 Azure Active Directory 中的群組名稱。

```
SELECT * FROM pg_get_session_roles() AS sess_roles(name name, roleid integer);

name                   roleid
--------------------------------
my_aad:test_group_1   106204
my_aad:test_group_2   106205
my_aad:test_group_3   106206
my_aad:test_group_4   106207
my_aad:test_group_5   106208
```

下列範例會針對目前登入的 IAM 使用者所屬的每個 IAM 群組，傳回一個資料列。傳回的資料欄會轉換為 sess_roles，具有欄 name 和 roleid。每個 name 都包含 IAM 命名空間和 IAM 群組名稱。

```
SELECT * FROM pg_get_session_roles() AS sess_roles(name name, roleid integer);

name                   roleid
--------------------------------
IAM:myGroup           110332
```

PG_LAST_COPY_COUNT

傳回目前工作階段中執行的最後一個 COPY 命令所載入的列數。PG_LAST_COPY_COUNT 會更新為最後一個 COPY ID，這是啟動載入程序的最後一個 COPY 的查詢 ID (即使載入失敗也一樣)。當 COPY 命令啟動載入程序時，查詢 ID 和 COPY ID 會更新。

如果 COPY 因為語法錯誤或權限不足而失敗，COPY ID 不會更新，PG_LAST_COPY_COUNT 會傳回上一個 COPY 的計數。如果目前工作階段中未執行任何 COPY 命令，或最後一個 COPY 在載入期間失敗，PG_LAST_COPY_COUNT 會傳回 0。如需詳細資訊，請參閱 PG_LAST_COPY_ID。

語法

```
pg_last_copy_count()
```

傳回類型

傳回 BIGINT。

範例

下列查詢傳回目前工作階段中最後一個 COPY 命令所載入的列數。

```
select pg_last_copy_count();

pg_last_copy_count
--------------------
             192497
(1 row)
```

PG_LAST_COPY_ID

傳回目前工作階段中最近完成之 COPY 命令的查詢 ID。如果目前工作階段中未執行任何 COPY 命令，PG_LAST_COPY_ID 會傳回 -1。

當 COPY 命令啟動載入程序時，PG_LAST_COPY_ID 的值會更新。如果 COPY 因為載入資料無效而失敗，COPY ID 會更新，因此您可以在查詢 STL_LOAD_ERRORS 資料表時使用 PG_LAST_COPY_ID。如果 COPY 交易還原，COPY ID 不會更新。

如果 COPY 命令因為載入程序開始之前發生的錯誤而失敗，例如語法錯誤、存取錯誤、登入資料無效或權限不足，COPY ID 不會更新。如果 COPY 在分析壓縮步驟 (成功連線之後開始，但在資料載入之前) 期間失敗，COPY ID 不會更新。

語法

```
pg_last_copy_id()
```

傳回類型

傳回整數。

範例

下列查詢傳回目前工作階段中最後一個 COPY 命令的查詢 ID。

```
select pg_last_copy_id();

pg_last_copy_id
---------------
          5437
(1 row)
```

下列查詢會將 STL_LOAD_ERRORS 聯結至 STL_LOADERROR_DETAIL，以檢視目前工作階段中最近載入期間發生之錯誤的詳細資訊：

```
select d.query, substring(d.filename,14,20),
d.line_number as line,
substring(d.value,1,16) as value,
substring(le.err_reason,1,48) as err_reason
from stl_loaderror_detail d, stl_load_errors le
where d.query = le.query
and d.query = pg_last_copy_id();

 query |    substring      | line |  value   |                    err_reason
-------+-------------------+------+----------
+-------------------------------------------------
    558| allusers_pipe.txt |  251 | 251      | String contains invalid or unsupported
 UTF8 code
    558| allusers_pipe.txt |  251 | ZRU29FGR | String contains invalid or unsupported
 UTF8 code
    558| allusers_pipe.txt |  251 | Kaitlin  | String contains invalid or unsupported
 UTF8 code
    558| allusers_pipe.txt |  251 | Walter   | String contains invalid or unsupported
 UTF8 code
```

PG_LAST_UNLOAD_ID

傳回目前工作階段中最近完成之 UNLOAD 命令的查詢 ID。如果目前工作階段中未執行任何 UNLOAD 命令，PG_LAST_UNLOAD_ID 會傳回 -1。

當 UNLOAD 命令啟動載入程序時，PG_LAST_UNLOAD_ID 的值會更新。如果 UNLOAD 因為載入資料無效而失敗，UNLOAD ID 會更新，所以您可以使用 UNLOAD ID 來進一步調查。如果 UNLOAD 交易還原，UNLOAD ID 不會更新。

如果 UNLOAD 命令因為載入程序開始之前發生的錯誤而失敗，例如語法錯誤、存取錯誤、登入資料無效或權限不足，UNLOAD ID 不會更新。

語法

```
PG_LAST_UNLOAD_ID()
```

傳回類型

傳回整數。

範例

下列查詢傳回目前工作階段中最後一個 UNLOAD 命令的查詢 ID。

```
select PG_LAST_UNLOAD_ID();

PG_LAST_UNLOAD_ID
---------------
          5437
(1 row)
```

PG_LAST_QUERY_ID

傳回目前工作階段中最近完成之查詢的查詢 ID。如果目前工作階段中未執行任何查詢，PG_LAST_QUERY_ID 會傳回 -1。PG_LAST_QUERY_ID 不傳回只在領導者節點上執行之查詢的查詢 ID。如需詳細資訊，請參閱 僅限領導節點函數。

語法

```
pg_last_query_id()
```

傳回類型

傳回整數。

範例

下列查詢傳回目前工作階段中完成的最後一個查詢的 ID。

```
select pg_last_query_id();
```

結果如下。

```
pg_last_query_id
----------------
           5437
(1 row)
```

下列查詢傳回目前工作階段中最近完成之查詢的查詢 ID 和文字。

```
select query, trim(querytxt) as sqlquery
from stl_query
where query = pg_last_query_id();
```

結果如下。

```
query | sqlquery
------+-------------------------------------------------
 5437 | select name, loadtime from stl_file_scan where loadtime > 1000000;
(1 rows)
```

PG_LAST_UNLOAD_COUNT

傳回目前工作階段中完成的最後一個 UNLOAD 命令所卸載的列數。PG_LAST_UNLOAD_COUNT 會更新為最後一個 UNLOAD 的查詢 ID，即使操作失敗也一樣。UNLOAD 完成時，查詢 ID 會更新。如果 UNLOAD 因為語法錯誤或權限不足而失敗，PG_LAST_UNLOAD_COUNT 會傳回上一個 UNLOAD 的計數。如果目前工作階段中未完成任何 UNLOAD 命令，或最後一個 UNLOAD 在卸載操作期間失敗，PG_LAST_UNLOAD_COUNT 會傳回 0。

語法

```
pg_last_unload_count()
```

傳回類型

傳回 BIGINT。

範例

下列查詢傳回目前工作階段中最後一個 UNLOAD 命令所卸載的列數。

```
select pg_last_unload_count();

pg_last_unload_count
--------------------
              192497
(1 row)
```

SLICE_NUM 函數

傳回整數，對應於列的資料在叢集內所在的配量編號。SLICE_NUM 不接受參數。

語法

```
SLICE_NUM()
```

傳回類型

SLICE_NUM 函數傳回整數。

範例

下列範例顯示 EVENT 資料表的前十個 EVENT 列有哪些配量包含資料：

```
select distinct eventid, slice_num() from event order by eventid limit 10;

 eventid | slice_num
---------+-----------
       1 |         1
       2 |         2
       3 |         3
       4 |         0
       5 |         1
       6 |         2
       7 |         3
       8 |         0
       9 |         1
      10 |         2
(10 rows)
```

下列範例傳回代碼 (10000)，表示不含 FROM 陳述式的查詢在領導者節點上執行：

```
select slice_num();
slice_num
-----------
10000
(1 row)
```

USER

CURRENT_USER 的同義詞。請參閱CURRENT_USER。

ROLE_IS_MEMBER_OF

如果角色是另一個角色的成員，則傳回 true。超級使用者可以檢查所有角色的成員資格。具有 ACCESS SYSTEM TABLE 許可的普通使用者可以檢查所有使用者的成員資格。否則，一般使用者只能檢查他們有權存取的角色。如果提供的角色不存在或目前使用者無法存取該角色，Amazon Redshift 就會發生錯誤。

語法

```
role_is_member_of( role_name,  granted_role_name)
```

引數

role_name

角色的名稱。

granted_role_name

授與角色的名稱。

傳回類型

傳回 BOOLEAN。

範例

下列查詢確認該角色不是 role1 的成員，也不是 role2 的成員。

```
SELECT role_is_member_of('role1', 'role2');
```

```
 role_is_member_of
-------------------
             False
```

USER_IS_MEMBER_OF

如果使用者是角色或群組的成員，則傳回 true。超級使用者可以檢查所有使用者的成員資格。屬於 sys:secadmin 或 sys:superuser 角色成員的一般使用者可以檢查所有使用者的成員資格。否則，一般使用者只能檢查自己。如果提供的身分不存在或目前的使用者無法存取角色，Amazon Redshift 就會傳送錯誤訊息。

語法

```
user_is_member_of( user_name,  role_name | group_name)
```

引數

user_name

使用者的名稱。

role_name

角色的名稱。

group_name

群組名稱。

傳回類型

傳回 BOOLEAN。

範例

下列查詢確認使用者不是 role1 的成員。

```
SELECT user_is_member_of('reguser', 'role1');

 user_is_member_of
-------------------
           False
```

VERSION

VERSION 函數會傳回目前安裝之版本的詳細資訊，而具體的 Amazon Redshift 版本資訊放在最後。

> **Note**
>
> 這是領導者節點函數。此函數在參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視時會傳回錯誤。

語法

```
VERSION()
```

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列範例顯示目前叢集的叢集版本資訊：

```
select version();
```

```
 version
----------------------------------------------------------------------------------------------------------------

PostgreSQL 8.0.2 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 3.4.2 20041017 (Red
Hat 3.4.2-6.fc3), Redshift 1.0.12103
```

其中 `1.0.12103` 是叢集版本編號。

> **Note**
>
> 若要強制將您的叢集版本更新至最新，請調整您的維護時段。

保留字

以下是 Amazon Redshift 保留字的清單。您可將保留字用於分隔的分隔識別碼 (雙引號)。

> **Note**
>
> 雖然 START 和 CONNECT 不是保留字，但如果您在查詢中使用 START 和 CONNECT 做為資料表別名，請使用分隔識別碼或 AS，以避免在執行期失敗。

如需更多詳細資訊，請參閱 名稱與識別碼。

```
AES128
AES256
ALL
ALLOWOVERWRITE
ANALYSE
ANALYZE
AND
ANY
ARRAY
AS
ASC
AUTHORIZATION
AZ64
BACKUP
BETWEEN
BINARY
BLANKSASNULL
BOTH
BYTEDICT
BZIP2
CASE
CAST
CHECK
COLLATE
COLUMN
CONSTRAINT
CREATE
CREDENTIALS
CROSS
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURRENT_USER_ID
DEFAULT
```

```
DEFERRABLE
DEFLATE
DEFRAG
DELTA
DELTA32K
DESC
DISABLE
DISTINCT
DO
ELSE
EMPTYASNULL
ENABLE
ENCODE
ENCRYPT
ENCRYPTION
END
EXCEPT
EXPLICIT
FALSE
FOR
FOREIGN
FREEZE
FROM
FULL
GLOBALDICT256
GLOBALDICT64K
GRANT
GROUP
GZIP
HAVING
IDENTITY
IGNORE
ILIKE
IN
INITIALLY
INNER
INTERSECT
INTERVAL
INTO
IS
ISNULL
JOIN
LEADING
LEFT
```

```
LIKE
LIMIT
LOCALTIME
LOCALTIMESTAMP
LUN
LUNS
LZO
LZOP
MINUS
MOSTLY16
MOSTLY32
MOSTLY8
NATURAL
NEW
NOT
NOTNULL
NULL
NULLS
OFF
OFFLINE
OFFSET
OID
OLD
ON
ONLY
OPEN
OR
ORDER
OUTER
OVERLAPS
PARALLEL
PARTITION
PERCENT
PERMISSIONS
PIVOT
PLACING
PRIMARY
RAW
READRATIO
RECOVER
REFERENCES
REJECTLOG
RESORT
RESPECT
```

```
RESTORE
RIGHT
SELECT
SESSION_USER
SIMILAR
SNAPSHOT
SOME
SYSDATE
SYSTEM
TABLE
TAG
TDES
TEXT255
TEXT32K
THEN
TIMESTAMP
TO
TOP
TRAILING
TRUE
TRUNCATECOLUMNS
UNION
UNIQUE
UNNEST
UNPIVOT
USER
USING
VERBOSE
WALLET
WHEN
WHERE
WITH
WITHOUT
```

系統資料表和檢視參考

主題

系統資料表和檢視

Amazon Redshift 具有許多系統資料表和檢視，其中包含系統如何運作的相關資訊。您可以使用查詢任何其他資料庫資料表的方式，來查詢這些系統資料表和檢視。本節示範一些系統資料表查詢的例子並說明：

- 系統如何產生不同類型的資料表和檢視
- 您可以從這些資料表取得什麼類型的資訊
- 如何將 Amazon Redshift 系統資料表聯結至目錄資料表
- 如何管理不斷增多的系統資料表日誌檔案

某些系統表只能由 AWS 工作人員用於診斷目的。以下章節討論系統管理員或其他資料庫使用者可以查詢實用資訊的系統資料表。

> **Note**
>
> 自動或手動叢集備份 (快照) 不包括系統資料表。STL 系統檢視會保留 7 天的日誌歷史記錄。保留日誌不需要客戶執行任何動作，但如果您想要儲存日誌資料超過 7 天，則必須定期將其複製到其他資料表或卸載到 Amazon S3。

系統資料表和檢視的類型

系統資料表和檢視有以下幾種類型：

- SVV 檢視包含有關資料庫物件的資訊，以及對暫時性 STV 資料表的參考。
- SYS 檢視可用來監控已佈建叢集和無伺服器工作群組的查詢和工作負載使用情況。
- STL 檢視是從已經保存到磁碟的日誌所產生，提供系統的歷史記錄。
- STV 資料表是虛擬系統資料表，包含目前系統資料的快照。他們是建基於暫時性記憶體內資料，而不會保存到磁碟型日誌或一般資料表。
- SVCS 檢視可提供主要叢集與並行擴展叢集查詢的詳細資訊。
- SVL 檢視提供有關主要叢集上查詢的詳細資料。

系統資料表和檢視不會使用相同的一致性模型做為一般資料表。因此查詢這些資料表和檢視時需要注意此問題，特別是查詢 STV 資料表和 SVV 檢視時。例如一般資料表 t1 (包含欄 c1)，預期以下查詢不會傳回任何列：

```
select * from t1
where c1 > (select max(c1) from t1)
```

但是以下對系統資料表的查詢可能會傳回列：

```
select * from stv_exec_state
where currenttime > (select max(currenttime) from stv_exec_state)
```

此查詢可能會傳回列的原因是 currenttime 為暫時性，而查詢中的兩個參考在評估時可能不會傳回相同的值。

另一方面，下列查詢可能不會傳回任何列：

```
select * from stv_exec_state
```

```
where currenttime = (select max(currenttime) from stv_exec_state)
```

系統資料表和檢視中資料的可見性

系統資料表和檢視中資料的可見性分為兩個類別：使用者可查看和超級使用者可查看。

擁有超級使用者權限的使用者才可以查看「超級使用者可查看」類別裡資料表的資料。一般使用者可以查看「使用者可查看」資料表裡的資料。若要讓一般使用者存取超級使用者可見的資料表，請將該資料表的 SELECT 權限授與一般使用者。如需詳細資訊，請參閱 GRANT。

依預設，在大部份「使用者可查看」資料表中，一般使用者無法查看另一位使用者產生的列。如果一般使用者被賦予 SYSLOG ACCESS UN CLICTED，則該使用者可以看到使用者可見資料表中的所有資料列，包括其他使用者產生的資料列。如需詳細資訊，請參閱 ALTER USER 或 CREATE USER 。所有使用者皆可看到 SVV_TRANSACTIONS 中的所有列。如需有關資料可見性的詳細資訊，請參閱 AWS re:Post 知識庫文章如何允許 Amazon Redshift 資料庫一般使用者檢視叢集中其他使用者的系統資料表中的資料？ 。

對於中繼資料檢視，Amazon Redshift 不允許不受限制地授予系統日誌存取權的使用者能見度。

> **Note**
>
> 若使用者擁有不受限制的系統資料表存取權限，該使用者就能查看其他使用者產生的資料。例如，STL_QUERY 和 STL_QUERY_TEXT 包含 INSERT、UPDATE 和 DELETE 陳述式的全文，當中可能包含使用者產生的敏感資料。

超級使用者可以查看所有資料表中的所有列。若要讓一般使用者存取「超級使用者可查看」資料表，請在該資料表上將 SELECT 權限授予 GRANT 一般使用者。

篩選系統產生的查詢

查詢相關的系統資料表和檢視，例如 SVL_QUERY_SUMMARY、SVL_QLOG 等等，通常包含大量自動產生的陳述式，Amazon Redshift 會使用這些陳述式來監控資料庫的狀態。超級使用者可以看到這些系統產生的查詢，但這些查詢通常作用不大。從使用 `userid` 欄的系統資料表或系統檢視選取時，若要把這些查詢篩選掉，請在 WHERE 子句新增 `userid > 1` 條件。例如：

```
select * from svl_query_summary where userid > 1
```

將僅佈建的查詢移轉至 SYS 監視檢視查詢

從已佈建的叢集遷移至 Amazon Redshift Serverless

如果您要將佈建的叢集遷移到 Amazon Redshift 無伺服器，則可能會使用下列系統檢視進行查詢，這些檢視只會儲存來自已佈建叢集的資料。

- 所有 STL 檢視
- 所有 STV 檢視
- 所有 SVCS 檢視
- 所有 SVL 檢視
- 一些 SVV 檢視
  - 如需 Amazon Redshift 無伺服器中不支援的 SVV 檢視完整清單，請參閱 Amazon Redshift 管理指南中的使用 Amazon Redshift 無伺服器監控查詢和工作負載底部的清單。

若要繼續使用您的查詢，請重新調整這些查詢，使其使用 SYS 監控檢視中定義的資料欄，這些欄對應於僅供佈建檢視中的欄。若要查看僅供佈建檢視與 SYS 監督視觀表之間的對應關係，請移至 用於移轉至 SYS 監視檢視的系統檢視對映

停留在已佈建叢集的同時更新查詢

如果您不是遷移到 Amazon Redshift Serverless，您可能仍想要更新現有的查詢。SYS 監控檢視的設計目的是易於使用和降低複雜性，提供完整的指標陣列以進行有效的監控和疑難排解。使用 SYS 檢視 (例如 SYS_QUERY_HISTORY 和 SYS_QUERY_DETAIL) 合併多個僅供佈建檢視的資訊，您可以簡化查詢。

使用 SYS 監控檢視改善查詢識別碼追蹤

SYS 監控檢視 (例如 SYS_QUERY_HISTORY 和 SYS_QUERY_DETAIL) 包含 query_id 欄，該欄保存使用者查詢的識別碼。同樣地，僅供佈建的檢視 (例如 STL_QUERY 和 SVL_QLOG) 包含查詢欄，此欄也包含查詢識別碼。不過，SYS 系統檢視中記錄的查詢識別碼與僅供佈建檢視中記錄的查詢識別碼不同。

SYS 檢視表的 query_id 欄值與僅供佈建檢視的查詢欄值之間的差異如下：

- 在 SYS 檢視中，query_id 欄會以其原始形式記錄使用者提交的查詢。Amazon Redshift 最佳化工具可能會將它們分解為子查詢以提高效能，但是您執行的單一查詢在 SYS_QUERY_HISTORY 中仍只有一列。如果您想查看個別子查詢，可以在SYS_QUERY_DETAIL中找到它們。
- 在僅佈建的檢視中，查詢欄會記錄子查詢層級的查詢。如果 Amazon Redshift 最佳化工具將您的原始查詢重寫為多個子查詢，對於您執行的單一查詢，STL_QUERY 中將有多列具有不同的查詢識別碼值。

當您將監控和診斷查詢從僅佈建檢視移轉至 SYS 檢視時，請考慮此差異，然後相應地編輯您的查詢。如需 Amazon Redshift 如何處理查詢的相關資訊，請參閱查詢計劃和執行工作流程。

範例

如需 Amazon Redshift 如何在僅佈建和 SYS 監控檢視中以不同方式記錄查詢的範例，請參閱下列查詢範例。這是按照您在 Amazon Redshift 中執行的方式所編寫的查詢。

```
SELECT
  s_name
  , COUNT(*) AS numwait
FROM
  supplier,
  lineitem l1,
  orders,
  nation
WHERE    s_suppkey = l1.l_suppkey
         AND o_orderkey = l1.l_orderkey
         AND o_orderstatus = 'F'
         AND l1.l_receiptdate > l1.l_commitdate
         AND EXISTS (SELECT
                       *
                     FROM
                       lineitem l2
                     WHERE  l2.l_orderkey = l1.l_orderkey
                            AND l2.l_suppkey <> l1.l_suppkey )
         AND NOT EXISTS (SELECT
                           *
                         FROM
                           lineitem l3
                         WHERE  l3.l_orderkey = l1.l_orderkey
                                AND l3.l_suppkey <> l1.l_suppkey
                                AND l3.l_receiptdate > l3.l_commitdate )
         AND s_nationkey = n_nationkey
```

```
         AND n_name = 'UNITED STATES'
GROUP BY
  s_name
ORDER BY
  numwait DESC
  , s_name LIMIT 100;
```

在此之下，Amazon Redshift 查詢最佳化工具會將上述使用者提交的查詢重寫為 5 個子查詢。

第一個子查詢建立一個暫存資料表來實現一個子查詢。

```
CREATE TEMP TABLE volt_tt_606590308b512(l_orderkey
                                       , l_suppkey
                                       , s_name   ) AS SELECT
                                                      l1.l_orderkey
                                                      , l1.l_suppkey
                                                      , public.supplier.s_name
                                                    FROM
                                                      public.lineitem AS l1,
                                                      public.nation,
                                                      public.orders,
                                                      public.supplier
                                                    WHERE  l1.l_commitdate <
 l1.l_receiptdate
                                                           AND l1.l_orderkey =
 public.orders.o_orderkey
                                                           AND l1.l_suppkey =
 public.supplier.s_suppkey
                                                           AND public.nation.n_name
 = 'UNITED STATES'::CHAR(8)
                                                           AND
 public.nation.n_nationkey = public.supplier.s_nationkey
                                                           AND
 public.orders.o_orderstatus = 'F'::CHAR(1);
```

第二個子查詢會從暫存資料表收集統計資料。

```
padb_fetch_sample: select count(*) from volt_tt_606590308b512;
```

第三個子查詢建立另一個暫存資料表來具體化另一個子查詢，引用上面建立的暫存資料表。

```
CREATE TEMP TABLE volt_tt_606590308c2ef(l_orderkey
```

```
                                        , l_suppkey) AS (SELECT

 volt_tt_606590308b512.l_orderkey
                                                          ,
 volt_tt_606590308b512.l_suppkey
                                                        FROM
                                                          public.lineitem AS l2,
                                                          volt_tt_606590308b512
                                                        WHERE  l2.l_suppkey <>
 volt_tt_606590308b512.l_suppkey
                                                               AND l2.l_orderkey =
 volt_tt_606590308b512.l_orderkey)
                                                               EXCEPT distinct (SELECT
 volt_tt_606590308b512.l_orderkey, volt_tt_606590308b512.l_suppkey
                                                               FROM public.lineitem AS
 l3, volt_tt_606590308b512
                                                               WHERE l3.l_commitdate <
 l3.l_receiptdate
                                                                 AND l3.l_suppkey <>
 volt_tt_606590308b512.l_suppkey
                                                                 AND l3.l_orderkey =
 volt_tt_606590308b512.l_orderkey);
```

第四個子查詢再次收集暫存資料表的統計資料。

```
padb_fetch_sample: select count(*) from volt_tt_606590308c2ef
```

最後一個子查詢使用上面建立的暫存資料表來產生輸出。

```
SELECT
  volt_tt_606590308b512.s_name AS s_name
  , COUNT(*) AS numwait
FROM
  volt_tt_606590308b512,
  volt_tt_606590308c2ef
WHERE    volt_tt_606590308b512.l_orderkey = volt_tt_606590308c2ef.l_orderkey
         AND volt_tt_606590308b512.l_suppkey = volt_tt_606590308c2ef.l_suppkey
GROUP BY
  1
ORDER BY
  2 DESC
  , 1 ASC LIMIT 100;
```

在僅佈建的系統檢視 STL_QUERY 中，Amazon Redshift 會在子查詢層級記錄五個資料列，如下所示：

```
SELECT userid, xid, pid, query, querytxt::varchar(100);
FROM stl_query
WHERE xid = 48237350
ORDER BY xid, starttime;

 userid |   xid    |    pid     |  query   |
    querytxt
--------+----------+------------+----------
+-----------------------------------------------------------------------------------------------------
    101 | 48237350 | 1073840810 | 12058151 | CREATE TEMP TABLE
 volt_tt_606590308b512(l_orderkey, l_suppkey, s_name) AS SELECT l1.l_orderkey, l1.l
    101 | 48237350 | 1073840810 | 12058152 | padb_fetch_sample: select count(*) from
 volt_tt_606590308b512
    101 | 48237350 | 1073840810 | 12058156 | CREATE TEMP TABLE
 volt_tt_606590308c2ef(l_orderkey, l_suppkey) AS (SELECT volt_tt_606590308b512.l_or
    101 | 48237350 | 1073840810 | 12058168 | padb_fetch_sample: select count(*) from
 volt_tt_606590308c2ef
    101 | 48237350 | 1073840810 | 12058170 | SELECT s_name , COUNT(*) AS numwait FROM
 supplier, lineitem l1, orders, nation WHERE s_suppkey = l1.
(5 rows)
```

在 SYS 監控檢視 SYS_QUERY_HISTORY 中，Amazon Redshift 會記錄查詢，如下所示：

```
SELECT user_id, transaction_id, session_id, query_id, query_text::varchar(100)
FROM sys_query_history
WHERE transaction_id = 48237350
ORDER BY start_time;

 user_id | transaction_id | session_id | query_id |
          query_text
---------+----------------+------------+----------
+-----------------------------------------------------------------------------------------------------
     101 |       48237350 | 1073840810 | 12058149 | SELECT s_name , COUNT(*) AS numwait
 FROM supplier, lineitem l1, orders, nation WHERE s_suppkey = l1.
```

在 SYS_QUERY_DETAIL 中，您可以使用來自 SYS_QUERY_HISTORY 的 query_id 值來尋找子項查詢層級的詳細資訊。child_query_sequence 欄顯示子查詢的執行順序。如需 SYS_QUERY_DETAIL 中欄的相關資訊，請參閱SYS_QUERY_DETAIL。

```
select user_id,
       query_id,
       child_query_sequence,
       stream_id,
       segment_id,
       step_id,
       start_time,
       end_time,
       duration,
       blocks_read,
       blocks_write,
       local_read_io,
       remote_read_io,
       data_skewness,
       time_skewness,
       is_active,
       spilled_block_local_disk,
       spilled_block_remote_disk
from sys_query_detail
where query_id = 12058149
      and step_id = -1
order by query_id,
         child_query_sequence,
         stream_id,
         segment_id,
         step_id;

 user_id | query_id | child_query_sequence | stream_id | segment_id | step_id |
       start_time         |          end_time          | duration | blocks_read |
 blocks_write | local_read_io | remote_read_io | data_skewness | time_skewness |
 is_active | spilled_block_local_disk | spilled_block_remote_disk
---------+----------+----------------------+-----------+------------+---------
+----------------------------+----------------------------+----------+-------------
+--------------+---------------+----------------+---------------+--------------
+-----------+--------------------------+--------------------------
     101 | 12058149 |                    1 |         0 |          0 |      -1 |
 2023-09-27 15:40:38.512415 | 2023-09-27 15:40:38.533333 |    20918 |           0 |
         0 |             0 |              0 |             0 |            44 | f
 |                         0 |                           0
     101 | 12058149 |                    1 |         1 |          1 |      -1 |
 2023-09-27 15:40:39.931437 | 2023-09-27 15:40:39.972826 |    41389 |          12 |
         0 |            12 |              0 |             0 |            77 | f
 |                         0 |                           0
```

```
    101 | 12058149 |                    1 |         2 |          2 |      -1 |
2023-09-27 15:40:40.584412 | 2023-09-27 15:40:40.613982 |    29570 |          32 |
        0 |            32 |              0 |             0 |           25 | f
|                         0 |                           0
    101 | 12058149 |                    1 |         2 |          3 |      -1 |
2023-09-27 15:40:40.582038 | 2023-09-27 15:40:40.615758 |    33720 |           0 |
        0 |             0 |              0 |             0 |            1 | f
|                         0 |                           0
    101 | 12058149 |                    1 |         3 |          4 |      -1 |
2023-09-27 15:40:46.668766 | 2023-09-27 15:40:46.705456 |    36690 |          24 |
        0 |            15 |              0 |             0 |           17 | f
|                         0 |                           0
    101 | 12058149 |                    1 |         4 |          5 |      -1 |
2023-09-27 15:40:46.707209 | 2023-09-27 15:40:46.709176 |     1967 |           0 |
        0 |             0 |              0 |             0 |           18 | f
|                         0 |                           0
    101 | 12058149 |                    1 |         4 |          6 |      -1 |
2023-09-27 15:40:46.70656  | 2023-09-27 15:40:46.71289  |     6330 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                         0 |                           0
    101 | 12058149 |                    1 |         5 |          7 |      -1 |
2023-09-27 15:40:46.71405  | 2023-09-27 15:40:46.714343 |      293 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                         0 |                           0
    101 | 12058149 |                    2 |         0 |          0 |      -1 |
2023-09-27 15:40:52.083907 | 2023-09-27 15:40:52.087854 |     3947 |           0 |
        0 |             0 |              0 |             0 |           35 | f
|                         0 |                           0
    101 | 12058149 |                    2 |         1 |          1 |      -1 |
2023-09-27 15:40:52.089632 | 2023-09-27 15:40:52.091129 |     1497 |           0 |
        0 |             0 |              0 |             0 |           11 | f
|                         0 |                           0
    101 | 12058149 |                    2 |         1 |          2 |      -1 |
2023-09-27 15:40:52.089008 | 2023-09-27 15:40:52.091306 |     2298 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                         0 |                           0
    101 | 12058149 |                    3 |         0 |          0 |      -1 |
2023-09-27 15:40:56.882013 | 2023-09-27 15:40:56.897282 |    15269 |           0 |
        0 |             0 |              0 |             0 |           29 | f
|                         0 |                           0
    101 | 12058149 |                    3 |         1 |          1 |      -1 |
2023-09-27 15:40:59.718554 | 2023-09-27 15:40:59.722789 |     4235 |           0 |
        0 |             0 |              0 |             0 |           13 | f
|                         0 |                           0
```

```
    101 | 12058149 |                    3 |         2 |          2 |      -1 |
2023-09-27 15:40:59.800382 | 2023-09-27 15:40:59.807388 |     7006 |           0 |
        0 |             0 |              0 |             0 |           58 | f
|                         0 |                           0
    101 | 12058149 |                    3 |         3 |          3 |      -1 |
2023-09-27 15:41:06.488685 | 2023-09-27 15:41:06.493825 |     5140 |           0 |
        0 |             0 |              0 |             0 |           56 | f
|                         0 |                           0
    101 | 12058149 |                    3 |         3 |          4 |      -1 |
2023-09-27 15:41:06.486206 | 2023-09-27 15:41:06.497756 |    11550 |           0 |
        0 |             0 |              0 |             0 |            2 | f
|                         0 |                           0
    101 | 12058149 |                    3 |         4 |          5 |      -1 |
2023-09-27 15:41:06.499201 | 2023-09-27 15:41:06.500851 |     1650 |           0 |
        0 |             0 |              0 |             0 |           15 | f
|                         0 |                           0
    101 | 12058149 |                    3 |         4 |          6 |      -1 |
2023-09-27 15:41:06.498609 | 2023-09-27 15:41:06.500949 |     2340 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                         0 |                           0
    101 | 12058149 |                    3 |         5 |          7 |      -1 |
2023-09-27 15:41:06.502945 | 2023-09-27 15:41:06.503282 |      337 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                         0 |                           0
    101 | 12058149 |                    4 |         0 |          0 |      -1 |
2023-09-27 15:41:06.62899  | 2023-09-27 15:41:06.631452 |     2462 |           0 |
        0 |             0 |              0 |             0 |           22 | f
|                         0 |                           0
    101 | 12058149 |                    4 |         1 |          1 |      -1 |
2023-09-27 15:41:06.632313 | 2023-09-27 15:41:06.63391  |     1597 |           0 |
        0 |             0 |              0 |             0 |           20 | f
|                         0 |                           0
    101 | 12058149 |                    4 |         1 |          2 |      -1 |
2023-09-27 15:41:06.631726 | 2023-09-27 15:41:06.633813 |     2087 |           0 |
        0 |             0 |              0 |             0 |            0 | f
|                         0 |                           0
    101 | 12058149 |                    5 |         0 |          0 |      -1 |
2023-09-27 15:41:12.571974 | 2023-09-27 15:41:12.584234 |    12260 |           0 |
        0 |             0 |              0 |             0 |           39 | f
|                         0 |                           0
    101 | 12058149 |                    5 |         0 |          1 |      -1 |
2023-09-27 15:41:12.569815 | 2023-09-27 15:41:12.585391 |    15576 |           0 |
        0 |             0 |              0 |             0 |            4 | f
|                         0 |                           0
```

```
    101 | 12058149 |                    5 |         1 |          2 |      -1 |
 2023-09-27 15:41:13.758513 | 2023-09-27 15:41:13.76401  |     5497 |           0 |
         0 |             0 |              0 |             0 |           39 | f
 |                         0 |                           0
    101 | 12058149 |                    5 |         1 |          3 |      -1 |
 2023-09-27 15:41:13.749    | 2023-09-27 15:41:13.772987 |    23987 |           0 |
         0 |             0 |              0 |             0 |           32 | f
 |                         0 |                           0
    101 | 12058149 |                    5 |         2 |          4 |      -1 |
 2023-09-27 15:41:13.799526 | 2023-09-27 15:41:13.813506 |    13980 |           0 |
         0 |             0 |              0 |             0 |           62 | f
 |                         0 |                           0
    101 | 12058149 |                    5 |         2 |          5 |      -1 |
 2023-09-27 15:41:13.798823 | 2023-09-27 15:41:13.813651 |    14828 |           0 |
         0 |             0 |              0 |             0 |            0 | f
 |                         0 |                           0
(28 rows)
```

系統資料表查詢、處理程序和無關 ID

分析顯示在系統資料表中的查詢、處理程序和工作階段 ID 時，請注意下列事項：

- 查詢 ID 值 (在欄中，如`query_id`和`query`) 可以隨著時間的推移重複使用。
- 進程 ID 或會話 ID 值（在諸如`process_id`，`pid`和的列中`session_id`）可以隨著時間的推移重複使用。
- 交易 ID 值 (在欄中，如`transaction_id`和`xid`) 是唯一的。

SVV 中繼資料檢視

SVV 檢視是 Amazon Redshift 中的系統檢視，其中包含資料庫物件的相關資訊。

> **Note**
>
> 如果資料庫回應因任何原因而失敗，Amazon Redshift 會報告 WARNING，而非ERROR。當您在資料共用中查詢物件時，Amazon Redshift 不會傳送 ERROR 訊息。

主題

- SVV_ACTIVE_CURSORS

- SVV_ALL_COLUMNS
- SVV_ALL_SCHEMAS
- SVV_ALL_TABLES
- SVV_ALTER_TABLE_RECOMMENDATIONS
- SVV_ATTACHED_MASKING_POLICY
- SVV_COLUMNS
- SVV_COLUMN_PRIVILEGES
- SVV_DATABASE_PRIVILEGES
- SVV_DATASHARE_PRIVILEGES
- SVV_DATASHARES
- SVV_DATASHARE_CONSUMERS
- SVV_DATASHARE_OBJECTS
- SVV_DEFAULT_PRIVILEGES
- SVV_DISKUSAGE
- SVV_EXTERNAL_COLUMNS
- SVV_EXTERNAL_DATABASES
- SVV_EXTERNAL_PARTITIONS
- SVV_EXTERNAL_SCHEMAS
- SVV_EXTERNAL_TABLES
- SVV_FUNCTION_PRIVILEGES
- SVV_GEOGRAPHY_COLUMNS
- SVV_GEOMETRY_COLUMNS
- SVV_IAM_PRIVILEGES
- SVV_IDENTITY_PROVIDERS
- SVV_INTEGRATION
- SVV_INTEGRATION_TABLE_STATE
- SVV_INTERLEAVED_COLUMNS
- SVV_LANGUAGE_PRIVILEGES
- SVV_MASKING_POLICY

- SVV_ML_MODEL_INFO
- SVV_ML_MODEL_PRIVILEGES
- SVV_MV_DEPENDENCY
- SVV_MV_INFO
- SVV_QUERY_INFLIGHT
- SVV_QUERY_STATE
- SVV_REDSHIFT_COLUMNS
- SVV_REDSHIFT_DATABASES
- SVV_REDSHIFT_FUNCTIONS
- SVV_REDSHIFT_SCHEMA_QUOTA
- SVV_REDSHIFT_SCHEMAS
- SVV_REDSHIFT_TABLES
- SVV_RELATION_PRIVILEGES
- SVV_RLS_APPLIED_POLICY
- SVV_RLS_ATTACHED_POLICY
- SVV_RLS_POLICY
- SVV_RLS_RELATION
- SVV_ROLE_GRANTS
- SVV_ROLES
- SVV_SCHEMA_PRIVILEGES
- SVV_SCHEMA_QUOTA_STATE
- SVV_SYSTEM_PRIVILEGES
- SVV_TABLE_INFO
- SVV_TABLES
- SVV_TRANSACTIONS
- SVV_USER_GRANTS
- SVV_USER_INFO
- SVV_VACUUM_PROGRESS
- SVV_VACUUM_SUMMARY

SVV_ACTIVE_CURSORS

SVV_ACTIVE_CURSORS 顯示目前開啟之游標的詳細資訊。如需詳細資訊，請參閱 DECLARE。

所有使用者都可看見 SVV_ACTIVE_CURSORS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。使用者只能檢視該使用者所開啟的游標。超級使用者可檢視所有游標。

資料表欄

欄名稱	資料類型	描述
user_id	integer	建立游標的使用者 ID。
cursor_name	varchar(128)	游標的名稱。
transaction_id	bigint(128)	交易的 ID。
session_id	integer	具有作用中游標的處理程序的 ID。
declare_time	timestamp	宣告游標的時間。
total_bytes	bigint	游標結果集的大小 (以位元組為單位)。
total_rows	bigint	游標結果集中的列數。
fetched_rows	bigint	目前從游標結果集擷取的列數。
cursor_storage_limit_used_percent	integer	指標目前使用的磁碟空間百分比。

SVV_ALL_COLUMNS

使用 SVV_ALL_COLUMNS 可檢視來自 Amazon Redshift 資料表的欄聯集，如 SVV_REDSHIFT_COLUMNS 所示，以及所有外部資料表中所有外部欄的合併清單。如需 Amazon Redshift 欄的相關資訊，請參閱SVV_REDSHIFT_COLUMNS。

所有使用者都可看見 SVV_ALL_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	資料庫的名稱。
schema_name	varchar(128)	結構描述的名稱。
table_name	varchar(128)	資料表的名稱。
column_name	varchar(128)	欄位的名稱。
ordinal_position	integer	資料表中欄位的位置。
column_default	varchar(4000)	欄位的預設值。
is_nullable	varchar(3)	指出欄位是否可為 Null 的值。可能的值是 yes 和 no。
data_type	varchar(128)	欄的資料類型。
character_maximum_length	integer	欄位中的最大字元數。
numeric_precision	integer	數值精確度。
numeric_scale	integer	數值擴展。
remarks	varchar(256)	備註。

範例查詢

下列範例會傳回 SVV_ALL_COLUMNS 的輸出。

```
SELECT *
FROM svv_all_columns
WHERE database_name = 'tickit_db'
    AND TABLE_NAME = 'tickit_sales_redshift'
ORDER BY COLUMN_NAME,
```

```
    SCHEMA_NAME
LIMIT 5;

 database_name | schema_name |     table_name        | column_name | ordinal_position
 | column_default | is_nullable | data_type | character_maximum_length |
 numeric_precision | numeric_scale | remarks
 --------------+-------------+----------------------+-------------+-----------------
+---------------+-------------+-----------+--------------------------
+------------------+--------------+---------
   tickit_db   |    public   | tickit_sales_redshift |    buyerid  |        4         |
               |      NO     |  integer  |                           |         32
  |       0       |
   tickit_db   |    public   | tickit_sales_redshift | commission  |        9         |
               |     YES     |  numeric  |                           |          8
  | 2       |
   tickit_db   |    public   | tickit_sales_redshift |    dateid   |        7         |
               |      NO     |  smallint |                           |         16
  |       0       |
   tickit_db   |    public   | tickit_sales_redshift |   eventid   |        5         |
               |      NO     |  integer  |                           |         32
  |       0       |
   tickit_db   |    public   | tickit_sales_redshift |    listid   |        2         |
               |      NO     |  integer  |                           |         32
  |       0       |
```

SVV_ALL_SCHEMAS

使用 SVV_ALL_SCHEMA 來檢視 Amazon Redshift 結構描述的聯集，如 SVV_REDSHIFT_SCHEMAS 所示，以及所有資料庫中所有外部結構描述的合併清單。如需 Amazon Redshift 結構描述的相關資訊，請參閱SVV_REDSHIFT_SCHEMAS。

所有使用者都可看見 SVV_ALL_SCHEMAS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	結構描述所在之資料庫的名稱。

欄名稱	資料類型	描述
schema_name	varchar(128)	結構描述的名稱。
schema_owner	integer	結構描述擁有者的使用者 ID。如需使用者 ID 的詳細資訊，請參閱PG_USER_INFO。
schema_type	varchar(128)	結構描述的類型。可能的值是 external、local 和 shared schemas。
schema_acl	varchar(128)	字串，定義結構描述之指定使用者或使用者群組的許可。
source_database	varchar(128)	外部結構描述的來源資料庫名稱。
schema_option	varchar(256)	結構描述的選項。這是外部結構描述屬性。

範例查詢

下列範例會傳回 SVV_ALL_SCHEMAS 的輸出。

```
SELECT *
FROM svv_all_schemas
WHERE database_name = 'tickit_db'
ORDER BY database_name,
    SCHEMA_NAME;


 database_name |    schema_name     | schema_owner | schema_type | schema_acl |
 source_database | schema_option
---------------+--------------------+--------------+-------------+------------
+-----------------+---------------
   tickit_db   |       public       |       1      |   shared    |            |
         |
```

SVV_ALL_TABLES

使用 SVV_ALL_TABLES 來檢視 Amazon Redshift 資料表的聯集，如 SVV_REDSHIFT_TABLES 所示，以及來自所有外部結構描述之所有外部資料表的合併清單。如需有關 Amazon Redshift 資料表的資訊，請參閱SVV_REDSHIFT_TABLES。

所有使用者都可看見 SVV_ALL_TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	資料表所在之資料庫的名稱。
schema_name	varchar(128)	資料表的結構描述名稱。
table_name	varchar(128)	資料表的名稱。
table_acl	varchar(128)	字串，定義資料表之指定使用者或使用者群組的許可。
table_type	varchar(128)	資料表的類型。可能的值是 views、base tables、external tables 和 shared tables。
remarks	varchar(256)	備註。

範例查詢

下列範例會傳回 SVV_ALL_TABLES 的輸出。

```
SELECT *
FROM svv_all_tables
WHERE database_name = 'tickit_db'
ORDER BY TABLE_NAME,
    SCHEMA_NAME
LIMIT 5;
```

```
 database_name | schema_name |        table_name        | table_type | table_acl |
 remarks
---------------+-------------+--------------------------+------------+-----------
+---------
   tickit_db   |    public   | tickit_category_redshift |    TABLE   |           |
   tickit_db   |    public   |   tickit_date_redshift   |    TABLE   |           |
   tickit_db   |    public   |   tickit_event_redshift  |    TABLE   |           |
   tickit_db   |    public   | tickit_listing_redshift  |    TABLE   |           |
   tickit_db   |    public   |   tickit_sales_redshift  |    TABLE   |           |
```

如果 table_acl 值為 null，表示沒有明確授予對應資料表的存取權限。

SVV_ALTER_TABLE_RECOMMENDATIONS

記錄資料表目前的 Amazon Redshift Advisor 建議。此檢視會顯示所有資料表的建議，不論這些資料表是否定義為自動最佳化。若要檢視資料表是否已定義為自動最佳化，請參閱SVV_TABLE_INFO。項目只會針對目前工作階段資料庫中可見的資料表而顯示。在套用建議之後 (無論是由 Amazon Redshift 或您執行)，該建議將不再出現在檢視中。

只有超級使用者才能看到 SVV_ALTER_TABLE_RECOMMENDATIONS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	Description (描述)
type	character (30)	建議的類型。可能的值是 distkey 和 sortkey。
database	character (128)	資料庫名稱。
table_id	integer	資料表識別碼。
group_id	integer	一組建議的群組編號。應該套用一組中的所有建議，以查看最大收益。對於排序索引鍵建議，可能的值為 -1；對於分散索引鍵建議，可能的值為大於零的數字。

欄名稱	資料類型	Description (描述)
DDL	character (1024)	必須執行才能套用建議的 SQL 陳述式。
auto_eligible	character (1)	該值表示該建議是否符合 Amazon Redshift 自動執行的資格。如果值為 t，則指示為 true，如果 f 則為 false。

範例查詢

在下列範例中，結果中的列會顯示分散索引鍵和排序索引鍵的建議。這些列也會顯示建議是否符合 Amazon Redshift 自動套用的資格。

```
select type, database, table_id, group_id, ddl, auto_eligible
from svv_alter_table_recommendations;
```

```
type      | database | table_id | group_id | ddl

                  | auto_eligible
diststyle | db0      | 117884   | 2        | ALTER TABLE "sch"."dp21235_tbl_1" ALTER
DISTSTYLE KEY DISTKEY "c0"
                    | f
diststyle | db0      | 117892   | 2        | ALTER TABLE "sch"."dp21235_tbl_1" ALTER
DISTSTYLE KEY DISTKEY "c0"
                    | f
diststyle | db0      | 117885   | 1        | ALTER TABLE "sch"."catalog_returns"
ALTER DISTSTYLE KEY DISTKEY "cr_sold_date_sk", ALTER COMPOUND SORTKEY
("cr_sold_date_sk","cr_returned_time_sk") | t
sortkey   | db0      | 117890   | -1       | ALTER TABLE "sch"."customer_addresses"
ALTER COMPOUND SORTKEY ("ca_address_sk")
                     | t
```

SVV_ATTACHED_MASKING_POLICY

使用 SVV_ATTACHED_MASKING_POLICY 可檢視目前連線資料庫上連接政策的所有關係和角色/使用者。

只有超級使用者和具有 sys:secadmin 角色的使用者才能檢視 SVV_ATTACHED_MASKING_POLICY。一般使用者將看到 0 列。

資料表欄

欄名稱	資料類型	描述
policy_name	text	連接至資料表的遮罩政策的名稱。
schema_name	text	連接政策之資料表的結構描述。
table_name	text	連接政策的資料表名稱。
table_type	text	連接政策的資料表類型。
grantor	text	連接政策的使用者名稱。
grantee	text	連接政策的使用者/角色名稱。
grantee_type	text	grantee 的類型。這可以是 role、user 或 public。
priority	int	連接政策的優先順序。
input_columns	text	連接政策的輸入欄屬性。
output_columns	text	連接政策的輸入欄屬性。

內部函數

SVV_ATTACHED_MASKING_POLICY 支援以下內部功能：

mask_get_policy_for_role_on_column

取得套用至指定欄/角色配對的最高優先順序政策。

語法

```
mask_get_policy_for_role_on_column
```

```
                        (relschema,
                        relname,
                        colname,
                        rolename);
```

參數

relschema

政策所在之結構描述的名稱。

relname

政策所在的資料表的名稱。

colname

連接政策之欄的名稱。

rolename

政策連接的角色的名稱。

mask_get_policy_for_user_on_column

取得套用至指定欄/使用者配對的最高優先順序政策。

語法

```
mask_get_policy_for_user_on_column
                        (relschema,
                        relname,
                        colname,
                        username);
```

參數

relschema

政策所在之結構描述的名稱。

relname

政策所在的資料表的名稱。

colname

連接政策之欄的名稱。

rolename

政策連接的使用者名稱。

SVV_COLUMNS

使用 SVV_COLUMNS 來檢視本機和外部資料表和檢視之欄位的相關資訊 (包含近期繫結的檢視)。

所有使用者都可看見 SVV_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

SVV_COLUMNS 檢視會繫結來自 系統目錄資料表 (字首為 PG 的資料表) 和 SVV_EXTERNAL_COLUMNS 系統畫面的資料表中繼資料。此系統目錄會說明 Amazon Redshift 資料庫資料表。SVV_EXTERNAL_COLUMNS 會說明與 Amazon Redshift Spectrum 一同使用的外部資料表。

所有使用者可以查看系統目錄資料表中的所有列。一般使用者僅能夠看見他們獲授權可存取之外部資料表的 SVV_EXTERNAL_COLUMNS 檢視之欄定義。雖然一般使用者可以看見系統目錄資料表中的資料表中繼資料，如果他們擁有該資料表，或獲授予存取權，他們只能夠從使用者定義的資料表選取資料。

資料表欄

欄名稱	資料類型	描述
table_catalog	text	資料表所在之目錄的名稱。
table_schema	text	資料表的結構描述名稱。
table_name	text	表格的名稱。
column_name	text	欄位的名稱。
ordinal_position	int	資料表中欄位的位置。
column_default	text	欄位的預設值。
is_nullable	text	指出欄位是否可為 Null 的值。

欄名稱	資料類型	描述
data_type	text	欄的資料類型。
character_maximum_length	int	欄位中的最大字元數。
numeric_precision	int	數值精確度。如果 data_type 欄是數值，則此欄會傳回整個值中的有效位數。
numeric_precision_radix	int	數值精準度基數。如果 data_type 欄是數值，則此欄會傳回 numeric_ precision 和 numeric_scale 欄的基數。
numeric_scale	int	數值擴展。如果 data_type 欄是數值，則此欄會傳回十進位值中的有效位數。
datetime_precision	int	日期時間精準度。
interval_type	text	間隔類型。
interval_precision	text	間隔精準度。
character_set_catalog	text	字元集目錄。
character_set_schema	text	字元集結構描述。
character_set_name	text	字元集名稱。
collation_catalog	text	定序目錄。
collation_schema	text	定序結構描述。
collation_name	text	定序名稱。
domain_name	text	網域名稱。
remarks	text	備註。

SVV_COLUMN_PRIVILEGES

使用 SVV_COLUMN_PRIVILEGES 可檢視明確授予目前資料庫中使用者、角色和群組的欄許可。

下列使用者可以看見 SVV_COLUMN_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	指定關係所在的命名空間的名稱。
relation_name	text	關係的名稱。
column_name	text	欄位的名稱。
privilege_type	text	許可的類型。可能的值是 SELECT 或 UPDATE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。

範例查詢

下列範例顯示 SVV_COLUMN_PRIVILEGES 的結果。

```
SELECT
 namespace_name,relation_name,COLUMN_NAME,privilege_type,identity_name,identity_type
```

```
FROM svv_column_privileges WHERE relation_name = 'lineitem';

 namespace_name | relation_name | column_name | privilege_type | identity_name |
 identity_type
----------------+---------------+-------------+----------------+---------------
+----------------
    public      |   lineitem    | l_orderkey  |     SELECT     |    reguser    |
 user
    public      |   lineitem    | l_orderkey  |     SELECT     |     role1     |
 role
    public      |   lineitem    | l_partkey   |     SELECT     |    reguser    |
 user
    public      |   lineitem    | l_partkey   |     SELECT     |     role1     |
 role
```

SVV_DATABASE_PRIVILEGES

使用 SVV_DATABASE_PRIVILEGES 來檢視明確授予給 Amazon Redshift 叢集中的使用者、角色和群組的資料庫許可。

下列使用者可以看見 SVV_DATABASE_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
database_name	text	資料庫的名稱。
privilege_type	text	許可的類型。可能的值是 USAGE、CREATE 或 TEMP。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。

欄名稱	資料類型	描述
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_DATABASE_PRIVILEGES 的結果。

```
SELECT database_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_database_privileges
WHERE database_name = 'test_db';

 database_name | privilege_type | identity_name | identity_type | admin_option
---------------+----------------+---------------+---------------+--------------
     test_db   |     CREATE     |     reguser   |      user     |     False
     test_db   |     CREATE     |      role1    |      role     |     False
     test_db   |     TEMP       |      public   |      public   |     False
     test_db   |     TEMP       |      role1    |      role     |     False
```

SVV_DATASHARE_PRIVILEGES

使用 SVV_DATASHARE_PRIVILEGES 來檢視明確授予給 Amazon Redshift 叢集中的使用者、角色和群組的資料共用許可。

下列使用者可以看見 SVV_DATASHARE_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
datashare_name	text	資料共用的名稱。
privilege_type	text	許可的類型。可能的值是 ALTER 或 SHARE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_DATASHARE_PRIVILEGES 的結果。

```
SELECT datashare_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_datashare_privileges
WHERE datashare_name = 'demo_share';

 datashare_name | privilege_type |  identity_name | identity_type | admin_option
----------------+----------------+----------------+---------------+--------------
   demo_share   |     ALTER      |    superuser   |     user      |   False
   demo_share   |     ALTER      |    reguser     |     user      |   False
```

SVV_DATASHARES

使用 SVV_DATASHARES 來檢視在叢集上建立的資料共用清單，以及與叢集共用的資料共用。

下列使用者可以看見 SVV_DATASHARES：

- 超級使用者
- 資料共用擁有者
- 對資料共用具有 ALTER 或 USAGE 許可的使用者

其他使用者看不到任何列。如需 ALTER 和 USAGE 許可的詳細資訊，請參閱GRANT。

資料表欄

欄名稱	資料類型	描述
share_name	varchar(128)	資料共用的名稱。
share_id	integer	資料共用的 ID。
share_owner	integer	資料共用的擁有者。
source_database	varchar(128)	此資料共用的來源資料庫。
consumer_database	varchar(128)	從此資料共用建立的消費者資料庫。
share_type	varchar(8)	資料共用的類型。可能的值是 INBOUND 和 OUTBOUND。
createdate	沒有時區的時間戳記	建立資料共用的日期。
is_publicaccessible	boolean	屬性，指定資料共用是否可以共用至可公開存取的叢集。
share_acl	varchar(256)	字串，定義資料共用之指定使用者或使用者群組的許可。
producer_account	varchar(16)	資料共用生產者帳戶的 ID。
producer_namespace	varchar(64)	資料共用生產者叢集的唯一叢集識別碼。
managed_by	varchar(64)	指定管理資料命名之 AWS 服務的屬性。

使用須知

擷取其他中繼資料 — 使用`share_owner`欄中傳回的整數，您可以與 `usesysid` in 聯結以取SVL_USER_INFO得資料清單擁有者的相關資料。這包括名稱和其他屬性。

範例查詢

下列範例會傳回 SVV_DATASHARES 的輸出。

```
SELECT share_owner, source_database, share_type, is_publicaccessible
FROM svv_datashares
WHERE share_name LIKE 'tickit_datashare%'
AND source_database = 'dev';

  share_owner | source_database | share_type  | is_publicaccessible
--------------+-----------------+-------------+----------------------
     100      |      dev        |   OUTBOUND  |        True
(1 rows)
```

下列範例會傳回傳出資料共用之 SVV_DATASHARES 的輸出。

```
SELECT share_name, share_owner, btrim(source_database), btrim(consumer_database),
 share_type, is_publicaccessible, share_acl, btrim(producer_account),
 btrim(producer_namespace), btrim(managed_by) FROM svv_datashares WHERE share_type =
 'OUTBOUND';

   share_name   | share_owner | source_database | consumer_database | share_type |
 is_publicaccessible | share_acl | producer_account|         producer_namespace
   | managed_by
----------------+-------------+-----------------+-------------------
+------------+---------------------+-----------+-----------------
+--------------------------------------+------------
    salesshare  |      1      |       dev       |                   |  OUTBOUND  |
   True         |           |   123456789012  | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |

 marketingshare |      1      |       dev       |                   |  OUTBOUND  |
   True         |           |   123456789012  | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
```

下列範例會傳回傳入資料共用之 SVV_DATASHARES 的輸出。

```
SELECT share_name, share_owner, btrim(source_database), btrim(consumer_database),
 share_type, is_publicaccessible, share_acl, btrim(producer_account),
 btrim(producer_namespace), btrim(managed_by) FROM svv_datashares WHERE share_type =
 'INBOUND';
```

```
  share_name    | share_owner | source_database | consumer_database | share_type |
 is_publicaccessible | share_acl | producer_account |         producer_namespace
    | managed_by
---------------+-------------+-----------------+-------------------
+------------+---------------------+-----------+-----------------
+--------------------------------------+------------
  salesshare    |             |                 |                   |  INBOUND   |
   False         |           |  123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
 |
 marketingshare |             |                 |                   |  INBOUND   |
  False         |           |  123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
 ADX
```

SVV_DATASHARE_CONSUMERS

使用 SVV_DATASHARE_CONSUMERS 來檢視叢集上建立之資料共用的消費者清單。

下列使用者可以看見 SVV_DATASHARE_CONSUMERS：

- 超級使用者
- 資料共用擁有者
- 對資料共用具有 ALTER 或 USAGE 許可的使用者

其他使用者看不到任何列。如需 ALTER 和 USAGE 許可的詳細資訊，請參閱GRANT。

資料表欄

欄名稱	資料類型	描述
share_name	varchar(128)	資料共用的名稱。
consumer_account	varchar(16)	資料共用消費者的帳戶 ID。
consumer_namespace	varchar(64)	資料共用消費者叢集的唯一叢集識別碼。
share_date	沒有時區的時間戳記	共用資料共用的日期。

範例查詢

下列範例會傳回 SVV_DATASHARE_CONSUMERS 的輸出。

```
SELECT count(*)
FROM svv_datashare_consumers
WHERE share_name LIKE 'tickit_datashare%';

1
```

SVV_DATASHARE_OBJECTS

使用 SVV_DATASHARE_OBJECTS 可檢視叢集上建立或與叢集共用之所有資料共用中的物件清單。

所有使用者都可看見 SVV_DATASHARE_OBJECTS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

如需有關檢視資料共用清單的詳細資訊，請參閱 SVV_DATASHARES。

資料表欄

欄名稱	資料類型	描述
share_type	varchar(8)	所指定資料共用的類型。可能的值是 OUTBOUND 和 INBOUND。
share_name	varchar(128)	資料共用的名稱。
object_type	varchar(64)	所指定物件的類型。可能的值是 schemas、tables、views、late binding views、materialized views 以及 functions。
object_name	varchar(512)	物件的名稱。物件名稱會延伸至包含結構描述名稱，例如 schema1.t1。
producer_account	varchar(16)	資料共用生產者帳戶的 ID。

欄名稱	資料類型	描述
producer_namespace	varchar(64)	資料共用生產者叢集的唯一叢集識別碼。
include_new	boolean	屬性會指定是否要將指定結構描述中建立的任何未來資料表、檢視或 SQL 使用者定義函數 (UDF) 新增至資料共用。此參數僅與 OUTBOUND 資料共用相關，並且僅與資料共用中的結構描述類型相關。

範例查詢

下列範例會傳回 SVV_DATASHARE_OBJECTS 的輸出。

```
SELECT share_type,
    btrim(share_name)::varchar(16) AS share_name,
    object_type,
    object_name
FROM svv_datashare_objects
WHERE share_name LIKE 'tickit_datashare%'
AND object_name LIKE '%tickit%'
ORDER BY object_name
LIMIT 5;

 share_type |     share_name     | object_type |          object_name
------------+--------------------+-------------+--------------------------------
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_category_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_date_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_event_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_listing_redshift
 OUTBOUND   |  tickit_datashare  |    table    |  public.tickit_sales_redshift
```

```
SELECT * FROM SVV_DATASHARE_OBJECTS WHERE share_name like 'sales%';

share_type | share_name | object_type | object_name  | producer_account |
 producer_namespace          | include_new
```

```
-----------+------------+-------------+--------------+-----------------
+--------------------------------------+-------------
 OUTBOUND  | salesshare | schema      | public       | 123456789012     |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |      t
 OUTBOUND  | salesshare | table       | public.sales | 123456789012     |
 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
```

SVV_DEFAULT_PRIVILEGES

使用 SVV_DEFAULT_PRIVILEGES 可檢視使用者在 Amazon Redshift 叢集中可存取的預設權限。

下列使用者可以看見 SVV_DEFAULT_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到授予他們的預設許可。

資料表欄

欄名稱	資料類型	描述
schema_name	text	結構描述的名稱。
object_type	text	物件的類型。可能的值是 RELATION、FUNCTION 或 PROCEDURE。
owner_id	integer	擁有者 ID。可能的值是 user ID。
owner_name	text	擁有者的名稱。
owner_type	text	擁有者類型。可能的值是 user。
privilege_type	text	權限類型。可能的值是 INSERT、SELECT、UPDATE、DELETE、RULE、REFERENCES TRIGGER、DROP 和 EXECUTE。
grantee_id	integer	承授者 ID。可能的值是 user ID、role ID 和 group ID。
grantee_type	text	承授者類型。可能的值是 user、role 和 public。

欄名稱	資料類型	描述
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組類型來說，永遠是 false。

範例查詢

下列範例會傳回 SVV_DEFAULT_PRIVILEGES 的輸出。

```
SELECT * from svv_default_privileges;

 schema_name |    object_type    | owner_id | owner_name | owner_type | privilege_type
 | grantee_id | grantee_name | grantee_type | admin_option
-------------+-------------------+--------- +------------+------------+----------------
+------------+--------------+--------------+-------------+
   public    |     RELATION      |    106   |     u1     |    user    |     UPDATE
 |     107    |      u2      |     user     |      f      |
   public    |     RELATION      |    106   |     u1     |    user    |     SELECT
 |     107    |      u2      |     user     |      f      |
```

SVV_DISKUSAGE

Amazon Redshift 透過繫結 STV_TBL_PERM 和 STV_BLOCKLIST 資料表來建立 SVV_DISKUSAGE 系統畫面。SVV_DISKUSAGE 檢視包含資料庫中該資料表之資料分配的相關資訊。

使用彙總查詢與 SVV_DISKUSAGE (如下範例所示) 以判斷每個資料庫、資料表、分割或欄位所配置的磁碟區塊數。每個資料區塊都使用 1 MB。您也可以使用 STV_PARTITIONS 來檢視磁碟使用率的摘要資訊。

只有超級使用者才能看到 SVV_DISKUSAGE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> 只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
db_id	integer	資料庫 ID。
name	character (72)	資料表名稱.
分割	integer	配置至資料表的資料分割。
col	integer	欄位的以零為起始的索引。您建立的每個資料表皆附加三個隱藏欄位：INSERT_XID、DELETE_XID 及 ROW_ID (OID)。包含 3 個使用者定義欄位的資料表包含 6 個實際欄位，使用者定義的欄位的內部編號為 0、1 及 2。在此範例中，INSERT_XID、DELETE_XID 及 ROW_ID 欄位的編號分別為 3、4 及 5。
tbl	integer	表格 ID。
blocknum	integer	資料區塊的 ID。
num_values	integer	區塊所包含的值的數量。
minvalue	bigint	區塊所包含的值下限。
maxvalue	bigint	區塊所包含的值上限。
sb_pos	integer	磁碟上的超級區塊位置的內部識別碼。
pinned	integer	區塊是否固定至記憶體做為預載的一部分。0 = false；1 = true。預設為 false。
on_disk	integer	區塊是否已自動儲存於磁碟。0 = false；1 = true。預設為 false。
modified	integer	區塊是否已修改。0 = false；1 = true。預設為 false。
hdr_modified	integer	區塊標頭是否已修改。0 = false；1 = true。預設為 false。
unsorted	integer	區塊是否未排序。0 = false；1 = true。預設為 true。

欄名稱	資料類型	描述
tombstone	integer	供內部使用。
preferred_diskno	integer	區塊應處於開啟狀態的磁碟數量 (無論磁碟是否故障)。一旦磁碟修復，區塊將移回該磁碟。
temporary	integer	無論區塊是否包含暫存資料，例如來自暫存資料表或中繼查詢結果。0 = false；1 = true。預設為 false。
newblock	integer	指出區塊是否是新的 (true) 或不曾遞交至磁碟 (false)。0 = false；1 = true。

範例查詢

SVV_DISKUSAGE 在每個配置的磁碟區塊上包含一個資料列，因此選取所有資料列的查詢可能會傳回數量非常大的資料列。建議僅適用彙總查詢搭配 SVV_DISKUSAGE。

傳回於 USERS 資料表中欄位 6 (EMAIL 欄位) 配置的區塊數上限：

```
select db_id, trim(name) as tablename, max(blocknum)
from svv_diskusage
where name='users' and col=6
group by db_id, name;

db_id  | tablename | max
--------+-----------+-----
175857 | users     |   2
(1 row)
```

下列查詢會為名為 SALESNEW 的大型 10 個欄位的資料表中所有欄位傳回類似結果。(欄位 10 到 12 的最後三列會供隱藏的中繼資料欄位使用。)

```
select db_id, trim(name) as tablename, col, tbl, max(blocknum)
from svv_diskusage
where name='salesnew'
group by db_id, name, col, tbl
order by db_id, name, col, tbl;
```

```
db_id  | tablename  | col |  tbl   | max
--------+-----------+-----+--------+-----
175857 | salesnew   |   0 | 187605 | 154
175857 | salesnew   |   1 | 187605 | 154
175857 | salesnew   |   2 | 187605 | 154
175857 | salesnew   |   3 | 187605 | 154
175857 | salesnew   |   4 | 187605 | 154
175857 | salesnew   |   5 | 187605 |  79
175857 | salesnew   |   6 | 187605 |  79
175857 | salesnew   |   7 | 187605 | 302
175857 | salesnew   |   8 | 187605 | 302
175857 | salesnew   |   9 | 187605 | 302
175857 | salesnew   |  10 | 187605 |   3
175857 | salesnew   |  11 | 187605 |   2
175857 | salesnew   |  12 | 187605 | 296
(13 rows)
```

SVV_EXTERNAL_COLUMNS

使用 SVV_EXTERNAL_COLUMNS 來檢視外部資料表中欄位的詳細資訊。也可以使用 SVV_EXTERNAL_COLUMNS 進行跨資料庫查詢，以檢視使用者有權存取的未連線資料庫上的資料表中所有欄的詳細資訊。

所有使用者都可看見 SVV_EXTERNAL_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
redshift_database_name	text	本機 Amazon Redshift 資料庫的名稱。
結構描述名稱	text	外部資料表之 Amazon Redshift 外部結構描述的名稱。
資料表名稱	text	外部資料表名稱。
columnname	text	欄位的名稱。

欄名稱	資料類型	描述
external_type	text	欄的資料類型。
columnnum	integer	外部資料欄號碼，從 1 開始。
part_key	integer	索引鍵的順序 (如果欄位為分割區索引鍵)。如果資料欄不是分割區，值為 **0**。
is_nullable	text	定義一個資料欄是否可為 null。有些值是 `true`、`false` 或 "" 空字串，代表沒有資訊。

SVV_EXTERNAL_DATABASES

使用 SVV_EXTERNAL_DATABASES 來檢視外部資料庫的詳細資訊。

所有使用者都可看見 SVV_EXTERNAL_DATABASES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
eskind	integer	資料庫外部目錄的類型；**1** 表示資料目錄，**2** 表示 Hive 中繼存放區。
esoptions	text	資料庫所在的目錄詳細資訊。
databasename	text	外部目錄中資料庫之名稱。
location	text	資料庫的位置。
parameters	text	資料庫參數。

SVV_EXTERNAL_PARTITIONS

使用 SVV_EXTERNAL_PARTITIONS 來檢視外部資料表中分割區的詳細資訊。

所有使用者都可看見 SVV_EXTERNAL_PARTITIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
結構描述名稱	text	Amazon Redshift 外部結構描述的名稱，適用於內含指定分割區的外部資料表。
資料表名稱	text	外部資料表名稱。
values	text	分割區的值。
location	text	分割區的位置。欄大小限制為 128 個字元。將會截斷較長的值。
input_format	text	輸入格式。
output_format	text	輸出格式。
serialization_lib	text	序列化程式庫。
serde_parameters	text	SerDe 參數。
已壓縮	integer	此值指出分割區是否已壓縮；**1** 表示已壓縮，**0** 表示未壓縮。
parameters	text	分割區屬性。

SVV_EXTERNAL_SCHEMAS

使用 SVV_EXTERNAL_SCHEMAS 來檢視外部結構描述的相關資訊。如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

所有使用者都可看見 SVV_EXTERNAL_SCHEMAS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
esoid	oid	外部結構描述 ID。
eskind	smallint	外部結構描述的外部目錄類型：1 表示資料目錄，2 表示 Hive 中繼存放區，3 表示 Aurora PostgreSQL 或 Amazon RDS PostgreSQL 的聯合查詢，4 表示本機 Amazon Redshift 資料庫的結構描述，5 表示遠端 Amazon Redshift 資料庫的結構描述，6 表示系統資料表的結構描述，8 表示遠端 MySQL 資料庫的結構描述，9 表示 Amazon Kinesis Data Streams 的結構描述，10 表示 Amazon Managed Streaming for Apache Kafka 資料串流。
結構描述名稱	name	外部結構描述名稱。
esowner	integer	外部結構描述擁有者的使用者 ID。
databasename	text	外部資料庫名稱。
esoptions	text	外部結構描述選項。

範例

下列範例顯示外部結構描述的詳細資訊。

```
select * from svv_external_schemas;

esoid  | eskind | schemaname | esowner | databasename | esoptions

-------+--------+------------+---------+--------------
+-------------------------------------------------------------
100133 |      1 | spectrum   |     100 | redshift     |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

SVV_EXTERNAL_TABLES

使用 SVV_EXTERNAL_TABLES 檢視外部資料表的詳細資訊；如需詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。也可以使用 SVV_EXTERNAL_TABLES 進行跨資料庫查詢，以檢視使用者有權存取的未連線資料庫上所有資料表的中繼資料。

所有使用者都可看見 SVV_EXTERNAL_TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
redshift_database_name	text	本機 Amazon Redshift 資料庫的名稱。
結構描述名稱	text	外部資料表之 Amazon Redshift 外部結構描述的名稱。
資料表名稱	text	外部資料表名稱。
tabletype	text	資料表的類型。某些值為 TABLE、VIEW、MATERIALIZED VIEW 或代表沒有資訊的空字串 " "。
location	text	資料表的位置。
input_format	text	輸入格式
output_format	text	輸出格式。
serialization_lib	text	序列化程式庫。
serde_parameters	text	SerDe 參數。
已壓縮	integer	此值指出資料表是否已壓縮；**1** 表示已壓縮，**0** 表示未壓縮。

欄名稱	資料類型	描述
parameters	text	資料表屬性。

範例

下列範例顯示詳細資料 svv_external_tables，其具有聯合查詢所使用之外部結構描述上的述詞。

```
select schemaname, tablename from svv_external_tables where schemaname = 'apg_tpch';
schemaname  | tablename
------------+-----------
apg_tpch    | customer
apg_tpch    | lineitem
apg_tpch    | nation
apg_tpch    | orders
apg_tpch    | part
apg_tpch    | partsupp
apg_tpch    | region
apg_tpch    | supplier
(8 rows)
```

SVV_FUNCTION_PRIVILEGES

使用 SVV_FUNCTION_PRIVILEGES 來檢視明確授予目前資料庫中使用者、角色和群組的函數許可。

下列使用者可以看見 SVV_FUNCTION_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	指定函數所在的命名空間名稱。

欄名稱	資料類型	描述
function_name	text	函數的名稱。
argument_types	text	代表函數輸入引數類型的字串。
privilege_type	text	許可的類型。可能的值是 EXECUTE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_FUNCTION_PRIVILEGES 的結果。

```
SELECT
 namespace_name,function_name,argument_types,privilege_type,identity_name,identity_type,admin_o
 FROM svv_function_privileges
WHERE identity_name IN ('role1', 'reguser');

 namespace_name | function_name |       argument_types       | privilege_type |
 identity_name | identity_type | admin_option
----------------+---------------+----------------------------+----------------
+----------------+---------------+-------------
    public      | test_func1    | integer                    |    EXECUTE     |
 role1     |     role      |  False
    public      | test_func2    | integer, character varying |    EXECUTE     |
 reguser    |     user      |  False
```

SVV_GEOGRAPHY_COLUMNS

使用 SVV_GEOGRAPHY_COLUMNS 來檢視資料倉儲中 GEOGRAPHY 欄的清單。此欄清單包含來自資料共用的欄。

所有使用者都可看見 SVV_GEOGRAPHY_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
f_table_catalog	varchar(128)	包含 GEOGRAPHY 欄的資料表所在之資料庫的名稱。
f_table_schema	varchar(128)	含有 GEOGRAPHY 欄之資料表所在的結構描述名稱。
f_table_name	varchar(128)	GEOGRAPHY 欄所在之資料表的名稱。
f_geography_column	varchar(128)	GEOGRAPHY 欄的名稱。
coord_dimension	integer	GEOGRAPHY 資料的維度數。
srid	integer	GEOMETRY 資料的空間參考系統識別碼 (SRID)。
type	varchar(128)	空間地理資料類型名稱。

範例查詢

下列範例顯示 SVV_GEOGRAPHY_COLUMNS 的結果。

```
SELECT * FROM svv_geography_columns;

f_table_catalog  | f_table_schema  | f_table_name  | f_geography_column  |
 coord_dimension | srid |  type
```

```
-----------------+-----------------+--------------+--------------------
+-----------------+------+-------------
dev               | public          | spatial_test  | test_geography      | 2
    | 0    | GEOGRAPHY
```

SVV_GEOMETRY_COLUMNS

使用 SVV_GEOMETRY_COLUMNS 可檢視資料倉儲中的 GEOMETRY 欄清單。此欄清單包含來自資料共用的欄。

所有使用者都可看見 SVV_GEOMETRY_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
f_table_catalog	varchar(128)	包含 GEOMETRY 欄的資料表所在之資料庫的名稱。
f_table_schema	varchar(128)	含有 GEOMETRY 欄之資料表所在的結構描述名稱。
f_table_name	varchar(128)	GEOMETRY 欄所在之資料表的名稱。
f_geometry_column	varchar(128)	GEOMETRY 欄的名稱。
coord_dimension	integer	GEOMETRY 資料的維度數。
srid	integer	GEOMETRY 欄的空間參考系統識別碼 (SRID)。
type	varchar(128)	空間幾何類型名稱。

範例查詢

下列範例顯示 SVV_GEOMETRY_COLUMNS 的結果。

```
SELECT * FROM svv_geometry_columns;

f_table_catalog  | f_table_schema  | f_table_name  | f_geometry_column   |
 coord_dimension | srid |  type
-----------------+-----------------+---------------+--------------------
+-----------------+------+-------------
dev              | public          | accomodations | shape               | 2
    | 0    | GEOMETRY
dev              | public          | zipcode       | wkb_geometry        | 2
    | 0    | GEOMETRY
```

SVV_IAM_PRIVILEGES

使用 SVV_IAM_PRIVILEGES 可檢視明確授予使用者、角色和群組的 IAM 權限。

下列使用者可以看見 SVV_IAM_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取的項目。

資料表欄

欄名稱	資料類型	描述
iam_arn	text	命名空間的名稱。
command_type	text	權限類型。可能的值是 COPY、UNLOAD、CREATE MODEL 或 EXTERNAL FUNCTION。
identity_id	integer	身分 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分名稱。

欄名稱	資料類型	描述
identity_type	text	身分類型。可能的值是 user、role、group 或 public。

範例查詢

以下範例顯示 SVV_IAM_PRIVILEGES 的結果。

```
SELECT * from SVV_IAM_PRIVILEGES ORDER BY IDENTITY_ID;
       iam_arn        | command_type | identity_id | identity_name | identity_type
----------------------+--------------+-------------+---------------+---------------
 default-aws-iam-role | COPY         |           0 | public        | public
 default-aws-iam-role | UNLOAD       |           0 | public        | public
 default-aws-iam-role | CREATE MODEL |           0 | public        | public
 default-aws-iam-role | EXFUNC       |           0 | public        | public
 default-aws-iam-role | COPY         |         106 | u1            | user
 default-aws-iam-role | UNLOAD       |         106 | u1            | user
 default-aws-iam-role | CREATE MODEL |         106 | u1            | user
 default-aws-iam-role | EXFUNC       |         106 | u1            | user
 default-aws-iam-role | COPY         |      118413 | r1            | role
 default-aws-iam-role | UNLOAD       |      118413 | r1            | role
 default-aws-iam-role | CREATE MODEL |      118413 | r1            | role
 default-aws-iam-role | EXFUNC       |      118413 | r1            | role
(12 rows)
```

SVV_IDENTITY_PROVIDERS

SVV_IDENTITY_PROVIDERS 檢視會傳回身分提供者的名稱和其他屬性。如需如何建立身分提供者的相關資訊，請參閱CREATE IDENTITY PROVIDER。

只有超級使用者才能看到 SVV_IDENTITY_PROVIDERS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
uid	integer	已註冊身分提供者的唯一 ID。
name	text	身分提供者名稱。
type	text	身分提供者類型。
instanceid	text	相同類型的執行個體之間的唯一差異化因素。
namespc	text	身分提供者的命名空間前置詞。
params	text	具有身分提供者參數的 JSON 物件。
已啟用	bool	指出身分提供者是否已啟用。

範例查詢

若要檢視身分提供者內容，請在建立身分提供者之後執行如下查詢。

```
SELECT name, type, instanceid, namespc, params, enabled
FROM svv_identity_providers
ORDER BY 1;
```

範例輸出包含參數描述。

```
       name       | type  |              instanceid              | namespc |

                                      params

                     | enabled
------------------+-------+--------------------------------------+---------
+-----------------------------------------------------------------------------------------------------------
+---------
```

```
 rs5517_azure_idp | azure | e40d4bb2-7670-44ae-bfb8-5db013221d73 | abc     |
 {"issuer":"https://login.microsoftonline.com/e40d4bb2-7670-44ae-bfb8-5db013221d73/
v2.0", "client_id":"871c010f-5e61-4fb1-83ac-98610a7e9110", "client_secret":,
 "audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift", "https://
analysis.windows.net/powerbi/connector/AWSRDS"]} | t
(1 row)
```

SVV_INTEGRATION

SVV_INTEGRATION 會顯示整合組態的詳細資訊。

只有超級使用者才能看到 SVV_INTEGRATION。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

如需零 ETL 整合的相關資訊，請參閱使用零 ETL 整合。

資料表欄

欄名稱	資料類型	描述
integration_id	character (128)	與整合關聯的識別碼。
target_database	character (128)	Amazon Redshift 中接收整合資料的資料庫。
source	character (128)	整合的來源資料。可能的類型包括 MySQL 和 PostgreSQL 。
state	character (128)	整合的狀態。可能的值包括 PendingDbConnectState 、SchemaDiscoveryState 、CdcRefreshState 與 ErrorState 。
current_lag	bigint	整合的來源與目的地之間的目前延遲時間 (毫秒)。
last_replicated_checkpoint	character (128)	最後複寫的檢查點。
total_tables_replicated	integer	目前處於已複寫狀態的資料表總數。
total_tables_failed	integer	目前處於失敗狀態的資料表總數。

欄名稱	資料類型	描述
creation_time	timestamp	建立整合時的時間 (UTC)。它被定義為從集成創建目標數據庫的時間。

範例查詢

下列 SQL 指令會顯示目前定義的整合。

```
select * from svv_integration;

           integration_id              | target_database | source |      state
   | current_lag |      last_replicated_checkpoint     | total_tables_replicated |
 total_tables_failed |       creation_time
---------------------------------------+-----------------+--------+----------------
+-------------+--------------------------------------+-------------------------
+---------------------+--------------------------
  99108e72-1cfd-414f-8cc0-0216acefac77 |     perfdb      |  MySQL | CdcRefreshState |
  56606106  | {"txn_seq":9834,"txn_id":126597515} |            152          |
 0         | 2023-09-19 21:05:27.520299
```

SVV_INTEGRATION_TABLE_STATE

SVV_INTEGRATION_TABLE_STATE 會顯示資料表層級整合資訊的詳細資料。

只有超級使用者才能看到 SVV_INTEGRATION_TABLE_STATE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

如需詳細資訊，請參閱使用零 ETL 整合。

資料表欄

欄名稱	資料類型	描述
integration_id	character(128)	與整合關聯的識別碼。
target_database	character(128)	Amazon Redshift 資料庫的名稱。
schema_name	character(128)	Amazon Redshift 結構描述的名稱。

欄名稱	資料類型	描述
table_name	character(128)	資料表的名稱。
table_state	character(128)	資料表的狀態。可能值為 Synced、Failed、Deleted、ResyncRequired 和 ResyncInitiated 。
table_last_replicated_checkpoint	character(128)	目前同步的記錄座標。
reason	character(256)	最後狀態轉換的原因。常見的原因可能是資料表中不支援的資料類型、資料表沒有主索引鍵。若要進一步了解如何疑難排解常見問題，請參閱在 Amazon Redshift 中對零 ETL 整合進行疑難排解。
last_updated_timestamp	沒有時區的時間戳記	資料表上次更新時的時間 (UTC)。

範例查詢

下列 SQL 命令會顯示整合的記錄。

```
select * from svv_integration_table_state;

          integration_id              | target_database | schema_name |     table_name
   | Table_state  |table_last_replicated_checkpoint | reason | last_updated_timestamp
-------------------------------------+-----------------+-------------
+------------------+-------------+--------------------------------
+-------------------------------------
 4798e675-8f9f-4686-b05f-92c538e19629 |  sample_test2   |    sample   |
 SampleTestChannel |    Synced    |   {"txn_seq":3,"txn_id":3122}   |        |
 2023-05-12 12:40:30.656625
```

SVV_INTERLEAVED_COLUMNS

使用 SVV_INTERLEAVED_COLUMNS 檢視來協助判斷是否應使用 VACUUM REINDEX，為使用交錯排序索引鍵的資料表重新建立索引。如需如何判斷執行 VACUUM 的頻率與執行 VACUUM REINDEX 的時間之相關資訊，請參閱管理清空時間。

只有超級使用者才能看到 SVV_INTERLEAVED_COLUMNS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
tbl	integer	表格 ID。
col	integer	欄位的以零為起始的索引。
interleaved_skew	numeric 9,2)	比例會指出在資料表之交錯排序索引鍵欄位中存在的扭曲量。值 1.00 表示沒有扭曲，值越大表示扭曲越嚴重。您必須使用 VACUUM REINDEX 命令來為具有嚴重扭曲的資料表重新建立索引。
last_reindex	timestar	指定資料表前次執行 VACUUM REINDEX 的時間。如果某個資料表從未重建索引，或如果基礎系統日誌資料表 STL_VACUUM 自上次重建索引後已經過輪換，則此值為 NULL。

範例查詢

若要識別可能需要重新建立索引的資料表，請執行下列查詢。

```
select tbl as tbl_id, stv_tbl_perm.name as table_name,
col, interleaved_skew, last_reindex
from svv_interleaved_columns, stv_tbl_perm
where svv_interleaved_columns.tbl = stv_tbl_perm.id
and interleaved_skew is not null;

 tbl_id | table_name | col | interleaved_skew | last_reindex
--------+------------+-----+------------------+---------------------
 100068 | lineorder  |   0 |             3.65 | 2015-04-22 22:05:45
```

```
 100068 | lineorder  |   1 |            2.65 | 2015-04-22 22:05:45
 100072 | customer   |   0 |            1.65 | 2015-04-22 22:05:45
 100072 | lineorder  |   1 |            1.00 | 2015-04-22 22:05:45
(4 rows)
```

SVV_LANGUAGE_PRIVILEGES

使用 SVV_LANGUAGE_PRIVILEGES 來檢視明確授予目前資料庫中的使用者、角色和群組的語言許可。

下列使用者可以看見 SVV_LANGUAGE_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
language_name	text	語言的名稱。
privilege_type	text	許可的類型。可能的值是 USAGE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_LANGUAGE_PRIVILEGES 的結果。

```
SELECT language_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_language_privileges
WHERE identity_name IN ('role1', 'reguser');

 language_name | privilege_type | identity_name | identity_type | admin_option
---------------+----------------+---------------+---------------+--------------
   exfunc      |     USAGE      |    reguser    |     user      |    False
   exfunc      |     USAGE      |     role1     |     role      |    False
   plpythonu   |     USAGE      |    reguser    |     user      |    False
```

SVV_MASKING_POLICY

使用 SVV_MASKING_POLICY 來檢視在叢集上建立的所有遮罩政策。

只有超級使用者和具有 sys:secadmin 角色的使用者才能檢視 SVV_MASKING_POLICY。一般使用者將看到 0 列。

資料表欄

欄名稱	資料類型	描述
policy_database	text	建立遮罩政策所在的資料庫名稱。
policy_name	text	遮罩政策的名稱。
input_columns	text	在 CREATE POLICY 陳述式的 WITH 子句中提供的屬性。
policy_expression	text	政策中使用的遮罩運算式。
policy_modified_by	text	上次修改政策的使用者名稱。
policy_modified_time	timestamp	政策建立或上次修改的時間戳記。

SVV_ML_MODEL_INFO

關於機器學習模型目前狀態的狀態資訊。

所有使用者都可看見 SVV_ML_MODEL_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	char(128)	模型的資料庫。
schema_name	char(128)	模型的結構描述。
user_name	char(128)	模型的擁有者。
model_name	char(128)	模型的名稱。
life_cycle	char(20)	模型的生命週期狀態。
is_refreshable	integer	模型的狀態，指出如果訓練查詢中的原始資料表和欄仍然存在且使用者仍具有這些項目的許可，則模型是否可重新整理。可能的值是：1 (可重新整理) 和 0 (無法重新整理)。
model_state	char(128)	膜性的目前狀態。

範例查詢

下列查詢會顯示機器學習模型的目前狀態。

```
SELECT schema_name, model_name, model_state
FROM svv_ml_model_info;

 schema_name |        model_name            |             model_state
-------------+------------------------------+-----------------------------------
 public      | customer_churn_auto_model    | Train Model On SageMaker In Progress
 public      | customer_churn_xgboost_model | Model is Ready
(2 row)
```

SVV_ML_MODEL_PRIVILEGES

使用 SVV_ML_MODEL_PRIVILEGES 來檢視明確授予叢集中使用者、角色和群組的機器學習模型許可。

下列使用者可以看見 SVV_ML_MODEL_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	所指定機器學習模型所在的命名空間名稱。
model_name	text	機器學習模型的名稱。
模型版本	integer	模型的版本號碼。
privilege_type	text	許可的類型。可能的值是 EXECUTE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_ML_MODEL_PRIVILEGES 的結果。

```
SELECT
 namespace_name,model_name,model_version,privilege_type,identity_name,identity_type,admin_optio
 FROM svv_ml_model_privileges
WHERE model_name = 'test_model';

 namespace_name | model_name | model_version | privilege_type |  identity_name |
 identity_type | admin_option
----------------+------------+---------------+----------------+----------------
+---------------+--------------
      public    | test_model |       1       |    EXECUTE     |     reguser    |
 user      |    False
      public    | test_model |       1       |    EXECUTE     |     role1      |
 role      |    False
```

SVV_MV_DEPENDENCY

SVV_MV_DEPENDENCY 資料表顯示了 Amazon Redshift 中具體化視觀表對其他具體化視觀表的相依性。

如需具體化視觀表的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

所有使用者都可看見 SVV_MV_DEPENDENCY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	char(128)	包含所指定具體化視觀表的資料庫。
schema_name	char(128)	具體化視觀表的結構描述。
name	char(128)	具體化視觀表的名稱。
dependent_database_name	char(128)	此具體化視觀表相依的具體化視觀表資料庫。

欄名稱	資料類型	描述
dependent_schema_name	char(128)	此具體化視觀表所依賴的具體化視觀表結構描述。
dependent_name	char(128)	此具體化視觀表所依賴的具體化視觀表的名稱。

範例查詢

下列查詢會傳回輸出列，指出具體化視觀表 mv_over_foo 使用其定義中的具體化視觀表 mv_foo 做為相依性。

```
CREATE SCHEMA test_ivm_setup;
CREATE TABLE test_ivm_setup.foo(a INT);
CREATE MATERIALIZED VIEW test_ivm_setup.mv_foo AS SELECT * FROM test_ivm_setup.foo;
CREATE MATERIALIZED VIEW test_ivm_setup.mv_over_foo AS SELECT * FROM
 test_ivm_setup.mv_foo;

SELECT * FROM svv_mv_dependency;

 database_name | schema_name          | name        | dependent_database_name |
 dependent_schema_name     | dependent_name
---------------+----------------------+-------------+-------------------------
+---------------------------+----------
 dev           | test_ivm_setup       | mv_over_foo |                     dev |
 test_ivm_setup            | mv_foo
```

SVV_MV_INFO

對於每個具體化視觀表，SVV_MV_INFO 資料表都會包含一個列，顯示資料是否過時，以及狀態資訊。

如需具體化視觀表的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

所有使用者都可看見 SVV_MV_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	char(128)	包含具體化視觀表的資料庫。
schema_name	char(128)	資料庫的結構描述。
user_name	char(128)	擁有具體化視觀表的使用者。
name	char(128)	具體化視觀表名稱。
is_stale	char(1)	t 表示具體化視觀表已過時。「過時」的具體化視觀表是基底資料表已更新，但具體化視觀表尚未重新整理時的具體化視觀表。如果重新整理自上一次重新啟動以來都並未執行，這項資訊可能會不準確。
state	integer	具體化視觀表的狀態如下： • 0 - 重新整理時，會完全重新計算具體化視觀表。 • 1 - 具體化視觀表為累加式。 • 101 - 具體化視觀表因遭到卸除的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 102 - 具體化視觀表因為資料欄類型變更而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 103 - 具體化視觀表因為重新命名的資料表而無法重新整理。 • 104 - 具體化視觀表因為重新命名的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 105 - 具體化視觀表因為重新命名的結構描述而無法重新整理。

欄名稱	資料類型	描述
autorewrite	char(1)	t 表示具體化視觀表適用於自動重新寫入查詢。
autorefresh	char(1)	t 表示可以自動重新整理具體化視觀表。

範例查詢

若要檢視所有具體化視觀表的狀態，請執行下列查詢。

```
select * from svv_mv_info;
```

此查詢傳回下列範例輸出。

```
database_name |       schema_name       | user_name |   name   |  is_stale | state |
 autorefresh | autorewrite
--------------+-------------------------+-----------+----------+-----------+-------
+-------------+----------------
 dev          | test_ivm_setup          | catch-22  | mv       |   f       |     1 |
       1 |           0
 dev          | test_ivm_setup          | lotr      | old_mv   |   t       |     1 |
       0 |           1
```

SVV_QUERY_INFLIGHT

使用 SVV_QUERY_INFLIGHT 檢視來判斷正在資料庫中執行的查詢為何。此檢視會聯合 STV_INFLIGHT 至 STL_QUERYTEXT。SVV_QUERY_INFLIGHT 不會顯示僅限於領導節點的查詢。如需詳細資訊，請參閱 僅限領導節點函數。

所有使用者都可看見 SVV_QUERY_INFLIGHT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> 只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
分割	integer	查詢執行時所在的分割。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。您可以使用此資料欄來聯結至 STL_ERROR 資料表。
starttime	timestamp	查詢開始的時間。
suspended	integer	查詢是否已遭停用：**0** = false；**1** = true。
text	character(200)	查詢文字，以 200 個字元的方式增量。
sequence	integer	查詢陳述式區段的序列數。

範例查詢

以下範例輸出顯示目前執行的兩個查詢，SVV_QUERY_INFLIGHT 查詢本身和查詢 428，此在資料表中分為三個列。(在此範例輸出中開始時間和陳述式欄位遭到截斷。)

```
select slice, query, pid, starttime, suspended, trim(text) as statement, sequence
from svv_query_inflight
order by query, sequence;

slice|query| pid  |      starttime       |suspended| statement | sequence
-----+-----+------+----------------------+---------+-----------+---------
1012 | 428 | 1658 | 2012-04-10 13:53:... |       0 | select ...|    0
1012 | 428 | 1658 | 2012-04-10 13:53:... |       0 | enueid ...|    1
1012 | 428 | 1658 | 2012-04-10 13:53:... |       0 | atname,...|    2
1012 | 429 | 1608 | 2012-04-10 13:53:... |       0 | select ...|    0
(4 rows)
```

SVV_QUERY_STATE

使用 SVV_QUERY_STATE 來檢視目前執行中查詢之執行期的相關資訊。

SVV_QUERY_STATE 檢視包含 STV_EXEC_STATE 資料表的資料子集。

所有使用者都可看見 SVV_QUERY_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

> **Note**
>
> 只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
seg	integer	正在執行中的查詢區段數。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	正在執行中的查詢步驟數。步驟是查詢執行期的最小單位。每個步驟表示分散的工作單位，例如掃描資料表、傳回結果或排序資料。
maxtime	間隔	此步驟執行的時間量上限 (微秒)。
avgtime	間隔	此步驟執行的平均時間 (微秒)。
rows	bigint	執行中步驟產生的列數。
bytes	bigint	執行中步驟產生的位元組數。
cpu	bigint	供內部使用。

欄名稱	資料類型	描述
memory	bigint	供內部使用。
rate_row	double precision	自查詢開始以來的 R ows-per-second 比率，計算方式是將資料列加總，並除以從查詢開始到目前時間的秒數。
rate_byte	double precision	自查詢啟動以來的 B ytes-per-second 速率，計算方式是將位元組加總，除以從查詢開始到目前時間的秒數。
label	character(25)	查詢標籤：步驟的名稱，例如 `scan` 或 `sort`。
is_diskbased	character(1)	此查詢步驟是否以磁碟型操作方式執行：true (**t**) 或 false (**f**)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。
num_parts	integer	雜湊資料表在雜湊步驟期間已分割的分割區數。此欄中的正數並不表示雜湊步驟以磁碟式操作方式執行。檢查 IS_DISKBASED 欄位中的值，查看該雜湊步驟是否為磁碟型。
is_rrscan	character(1)	若為 true (**t**)，表示已在步驟上使用範圍限制掃描。預設為 false (**f**)。
is_delayed_scan	character(1)	若為 true (**t**)，表示已在該步驟上使用延遲的掃描。預設為 false (**f**)。

範例查詢

依步驟判斷查詢處理時間

下列查詢顯示查詢 ID 為 279 之查詢的每個步驟執行多久，以及 Amazon Redshift 處理的列多寡：

```
select query, seg, step, maxtime, avgtime, rows, label
from svv_query_state
where query = 279
order by query, seg, step;
```

此查詢會擷取查詢 279 的相關處理資訊，如下範例輸出所示：

```
query |   seg   | step | maxtime | avgtime |  rows   | label
------+---------+------+---------+---------+---------+------------------
  279 |       3 |    0 | 1658054 | 1645711 | 1405360 | scan
  279 |       3 |    1 | 1658072 | 1645809 |       0 | project
  279 |       3 |    2 | 1658074 | 1645812 | 1405434 | insert
  279 |       3 |    3 | 1658080 | 1645816 | 1405437 | distribute
  279 |       4 |    0 | 1677443 | 1666189 | 1268431 | scan
  279 |       4 |    1 | 1677446 | 1666192 | 1268434 | insert
  279 |       4 |    2 | 1677451 | 1666195 |       0 | aggr
(7 rows)
```

判斷磁碟上是否有任何正在執行的作用中查詢

以下查詢顯示作用中的查詢目前是否正在磁碟上執行：

```
select query, label, is_diskbased from svv_query_state
where is_diskbased = 't';
```

此範例輸出顯示作用中的查詢目前是否正在磁碟上執行：

```
 query | label        | is_diskbased
-------+--------------+--------------
1025   | hash tbl=142 |      t
(1 row)
```

SVV_REDSHIFT_COLUMNS

使用 SVV_REDSHIFT_COLUMNS 來檢視使用者有權存取的所有欄清單。這組欄包括叢集上的欄，以及遠端叢集所提供資料共用中的欄。

所有使用者都可看見 SVV_REDSHIFT_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	包含欄的資料表所在之資料庫的名稱。

欄名稱	資料類型	描述
schema_name	varchar(128)	資料表的結構描述名稱。
table_name	varchar(128)	資料表的名稱。
column_name	varchar(128)	欄的名稱。
ordinal_position	integer	資料表中欄位的位置。
data_type	varchar(32)	欄的資料類型。
column_default	varchar(4000)	欄位的預設值。
is_nullable	varchar(3)	定義欄是否可為 null 的值。可能的值是 yes、no 或 " " (表示沒有資訊的空字串)。
編碼	varchar(128)	欄的編碼類型。
distkey	boolean	如果此欄是資料表的分散索引鍵，則為 true，否則為 false。
sortkey	integer	指定排序索引鍵中欄順序的值。 如果資料表使用的是複合排序索引鍵，則排序索引鍵部分的所有欄位會有正值，指示排序索引鍵中欄位的位置。 如果資料表使用交錯排序索引鍵，那麼作為排序索引鍵一部分的每一欄都會有交替為正或負的值。在這裡，絕對值指出欄在排序索引鍵的位置。 若 sortkey 為 0，則欄不是排序索引鍵的一部分。

欄名稱	資料類型	描述
column_acl	varchar(128)	字串，定義欄之指定使用者或使用者群組的許可。
備註	varchar(256)	備註。

範例查詢

下列範例會傳回 SVV_REDSHIFT_COLUMNS 的輸出。

```
SELECT *
FROM svv_redshift_columns
WHERE database_name = 'tickit_db'
    AND TABLE_NAME = 'tickit_sales_redshift'
ORDER BY COLUMN_NAME,
    TABLE_NAME,
    database_name
LIMIT 5;

database_name | schema_name |       table_name      | column_name | ordinal_position |
 data_type | column_default | is_nullable | encoding | distkey | sortkey | column_acl
 | remarks
--------------+-------------+-----------------------+-------------+------------------
+-----------+----------------+-------------+----------+---------+---------
+-------------+--------
   tickit_db  |   public    | tickit_sales_redshift |   buyerid   |        4         |
 integer  |                |      NO     |   az64   |  False  |    0    |             |
   tickit_db  |   public    | tickit_sales_redshift |  commission |        9         |
 numeric  |      (8,2)     |     YES     |   az64   |  False  |    0    |             |
   tickit_db  |   public    | tickit_sales_redshift |    dateid   |        6         |
 smallint |                |      NO     |   none   |  False  |    1    |             |
   tickit_db  |   public    | tickit_sales_redshift |   eventid   |        5         |
 integer  |                |      NO     |   az64   |  False  |    0    |       |
   tickit_db  |   public    | tickit_sales_redshift |   listid    |        2         |
 integer  |                |      NO     |   az64   |  True   |    0    |             |
```

SVV_REDSHIFT_DATABASES

使用 SVV_ REDSHIFT_DATABASES 來檢視使用者有權存取的所有資料庫清單。這包括叢集上的資料庫，以及從遠端叢集所提供資料共用建立的資料庫。

所有使用者都可看見 SVV_REDSHIFT_DATABASES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	資料庫的名稱。
database_owner	integer	資料庫擁有者使用者 ID。
database_type	varchar(32)	資料庫的類型。可能的類型為本機或共用資料庫。
database_acl	varchar(128)	此資訊僅供內部使用。
database_options	varchar(128)	資料庫的屬性。
database_isolation_level	varchar(128)	資料庫的隔離層級。可能的值包括：Snapshot Isolation 和 Serializable 。

範例查詢

下列範例會傳回 SVV_REDSHIFT_DATABASES 的輸出。

```
select database_name, database_owner, database_type, database_options,
 database_isolation_level
from  svv_redshift_databases;


database_name | database_owner | database_type | database_options |
 database_isolation_level
--------------+----------------+---------------+------------------+------------------
   dev        |  1             | local         | NULL             | Serializable
```

SVV_REDSHIFT_FUNCTIONS

使用 SVV_REDSHIFT_FUNCTIONS 來檢視使用者有權存取的所有函數的清單。這組函數包括叢集上的函數，以及遠端叢集所提供資料共用中的函數。

所有使用者都可看見 SVV_REDSHIFT_FUNCTIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	具有這些函數之叢集所在的資料庫名稱。
schema_name	varchar(128)	指定給定函數之結構描述的名稱。
function_name	varchar(128)	所指定函數的名稱。
function_type	varchar(128)	函數的類型。可能的值是 regular functions、aggregate functions 和 stored procedures。
argument_type	varchar(512)	字串；代表函數輸入引數的類型。
result_type	varchar(128)	函數傳回值的資料類型。

範例查詢

下列範例會傳回 SVV_REDSHIFT_FUNCTIONS 的輸出。

```
SELECT *
FROM svv_redshift_functions
WHERE database_name = 'tickit_db'
    AND SCHEMA_NAME = 'public'
ORDER BY function_name
```

```
LIMIT 5;

database_name | schema_name |      function_name    |  function_type   |
 argument_type  | result_type
--------------+-------------+-----------------------+------------------
+------------------+-------------
   tickit_db  |    public   |     shared_function   | REGULAR FUNCTION | integer,
 integer |   integer
```

SVV_REDSHIFT_SCHEMA_QUOTA

顯示資料庫中每個結構描述的配額和目前磁碟使用量。

所有使用者都可看見 SVV_REDSHIFT_SCHEMA_QUOTA。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

查詢已佈建叢集或 Redshift 無伺服器工作群組時，可使用此檢視。

資料表欄

欄名稱	資料類型	描述
database_name	character(128)	包含結構描述的資料庫。
schema_name	character(128)	結構描述的名稱。
schema_owner	integer	結構描述擁有者的內部使用者 ID。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。

範例查詢

下列範例會顯示名為 `sales_schema` 之結構描述的配額和目前磁碟使用量。

```
SELECT TRIM(SCHEMA_NAME) "schema_name", QUOTA, disk_usage FROM
 svv_redshift_schema_quota
```

```
WHERE SCHEMA_NAME = 'sales_schema';


schema_name   | quota | disk_usage
--------------+-------+-----------
sales_schema  | 2048  | 30
```

SVV_REDSHIFT_SCHEMAS

使用 SVV_REDSHIFT_SCHEMAS 來檢視使用者有權存取之所有結構描述的清單。這組結構描述包括叢集上的結構描述，以及遠端叢集所提供之資料庫的結構描述。

所有使用者都可看見 SVV_REDSHIFT_SCHEMAS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	指定結構描述所在的資料庫名稱。
schema_name	varchar(128)	命名空間或結構描述名稱。
schema_owner	integer	結構描述擁有者的內部使用者 ID。
schema_type	varchar(16)	結構描述的類型。可能的值是 shared 和 local schemas。
schema_acl	varchar(128)	字串，定義結構描述之指定使用者或使用者群組的許可。
schema_option	varchar(128)	結構描述的選項。

範例查詢

下列範例會傳回 SVV_REDSHIFT_SCHEMAS 的輸出。

```
SELECT *
FROM svv_redshift_schemas
WHERE database_name = 'tickit_db'
ORDER BY database_name,
    SCHEMA_NAME;

database_name |    schema_name     | schema_owner | schema_type | schema_acl |
 schema_option
--------------+--------------------+--------------+-------------+------------
+---------------
   tickit_db  |       public       |       1      |    shared   |            |
```

SVV_REDSHIFT_TABLES

使用 SVV_REDSHIFT_TABLES 來檢視使用者有權存取的所有資料表清單。這組資料表包括叢集上的資料表，以及遠端叢集所提供資料共用中的資料表。

所有使用者都可看見 SVV_REDSHIFT_TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	指定之資料表所在資料庫的名稱。
schema_name	varchar(128)	資料表的結構描述名稱。
table_name	varchar(128)	資料表的名稱。
table_type	varchar(128)	資料表的類型。可能的值是 views 和 tables。
table_acl	varchar(128)	字串，定義資料表之指定使用者或使用者群組的許可。
備註	varchar(128)	備註。
table_owner	varchar(128)	資料表的擁有者。

範例查詢

下列範例會傳回 SVV_REDSHIFT_TABLES 的輸出。

```
SELECT *
FROM svv_redshift_tables
WHERE database_name = 'tickit_db' AND TABLE_NAME LIKE 'tickit_%'
ORDER BY database_name,
TABLE_NAME;

database_name | schema_name |         table_name       | table_type | table_acl |
 remarks | table_owner
--------------+-------------+--------------------------+------------+-----------
+---------+-----------
   tickit_db  |    public   | tickit_category_redshift |    TABLE   |           |
   +
   tickit_db  |    public   |   tickit_date_redshift   |    TABLE   |           |
   +
   tickit_db  |    public   |   tickit_event_redshift  |    TABLE   |           |
   +
   tickit_db  |    public   |  tickit_listing_redshift |    TABLE   |           |
   +
   tickit_db  |    public   |   tickit_sales_redshift  |    TABLE   |           |
   +
   tickit_db  |    public   |   tickit_users_redshift  |    TABLE   |           |
   +
   tickit_db  |    public   |   tickit_venue_redshift  |    TABLE   |           |
```

如果 table_acl 值為 null，表示沒有明確授予對應資料表的存取權限。

SVV_RELATION_PRIVILEGES

使用 SVV_RELATION_PRIVILEGES 來檢視明確授予目前資料庫中使用者、角色和群組的關係 (資料表和檢視) 許可。

下列使用者可以看見 SVV_RELATION_PRIVILEGES：

- 超級使用者
- 具有 SYSLOG 存取不受限制權限的使用者

其他使用者只能看到他們有權存取或擁有的身分。如需資料可見性的更多資訊，請參閱〈系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	指定關係所在的命名空間的名稱。
relation_name	text	關係的名稱。
privilege_type	text	許可的類型。可能的值是 INSERT、SELECT、UPDATE、DELETE、REFERENCES 或 DROP。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_RELATION_PRIVILEGES 的結果。

```
SELECT
 namespace_name,relation_name,privilege_type,identity_name,identity_type,admin_option
 FROM svv_relation_privileges
WHERE relation_name = 'orders' AND privilege_type = 'SELECT';

 namespace_name | relation_name | privilege_type |  identity_name | identity_type |
 admin_option
----------------+---------------+----------------+----------------+---------------
+--------------
     public     |    orders     |     SELECT     |    reguser     |     user      |
 False
     public     |    orders     |     SELECT     |     role1      |     role      |
 False
```

SVV_RLS_APPLIED_POLICY

使用 SVV_RLS_APPLIED_POLICY 可追蹤 RLS 政策在參考受 RLS 保護關係之查詢上的應用。

下列使用者可以看見 SVV_RLS_APPLIED_POLICY：

- 超級使用者
- 具有 `sys:operator` 角色的使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

請注意，sys:secadmin 未授予此系統許可。

資料表欄

欄名稱	資料類型	描述
使用者名稱	text	執行查詢的使用者名稱。
query	integer	查詢的 ID。
xid	long	交易的內容。
pid	integer	執行查詢的領導者程序。
recordtime	time	記錄查詢的時間。
command	char(1)	套用 RLS 政策的命令。可能的值包括：k 代表未知、s 代表選取、u 代表更新、i 代表插入、y 代表公用程式，以及 d 代表刪除。
datname	text	列層級安全政策所連接之關係的資料庫名稱。
relschema	text	列層級安全政策所連接之關係的結構描述名稱。
relname	text	列層級安全政策所連接的關係名稱。
polname	text	連接至關係的列層級安全政策名稱。
poldefault	char(1)	連接至關係之列層級安全政策的預設設定。可能的值為如果已套用預設 false 政策則為 f (代表 false)， 如果已套用預設 true 政策則為 t (代表 true)。

範例查詢

下列範例顯示 SVV_RLS_APPLIED_POLICY 的結果。若要查詢 SVV_RLS_APPLIED_POLICY，您必須具有 ACCESS SYSTEM TABLE 許可。

```
-- Check what RLS policies were applied to the run query.
SELECT username, command, datname, relschema, relname, polname, poldefault
FROM svv_rls_applied_policy
WHERE datname = CURRENT_DATABASE() AND query = PG_LAST_QUERY_ID();

 username | command |  datname  | relschema |          relname          |      polname
  | poldefault
----------+---------+-----------+-----------+---------------------------
+-----------------+------------
   molly  |    s    | tickit_db |   public  | tickit_category_redshift |
 policy_concerts |
```

SVV_RLS_ATTACHED_POLICY

使用 SVV_RLS_ATTACHED_POLICY 來檢視目前連線的資料庫上連接了一或多個列層級安全政策的所有關係和使用者清單。

只有具備 sys:secadmin 角色的使用者才能查詢此檢視。

資料表欄

欄名稱	資料類型	描述
relschema	text	列層級安全政策所連接之關係的結構描述名稱。
relname	text	列層級安全政策所連接的關係名稱。
relkind	text	物件的類型，例如 table。
polname	text	連接至關係之列層級安全政策的名稱。
grantor	text	連接此政策的使用者名稱。
grantee	text	此政策已連接之使用者或角色的名稱。
granteekind	text	grantee 的類型。可能的值是 user 或 role。

欄名稱	資料類型	描述
is_pol_on	boolean	指出資料表上的列層級安全政策是開啟或關閉的參數。可能的值是 true 和 false。
is_rls_on	boolean	指出資料表上的列層級安全是開啟或關閉的參數。可能的值是 true 和 false。
rls_conjunction_type	character (3)	指出關係是否與 and 或 or 結合 RLS 政策的參數。

範例查詢

下列範例顯示 SVV_RLS_ATTACHED_POLICY 的結果。

```
--Inspect the policy in SVV_RLS_ATTACHED_POLICY
SELECT * FROM svv_rls_attached_policy;

 relschema |        relname           | relkind |     polname     | grantor | grantee
 | granteekind | is_pol_on | is_rls_on | rls_conjuntion_type
-----------+--------------------------+---------+-----------------+---------+----------
+-------------+-----------+-----------+---------------------
 public    | tickit_category_redshift |  table  | policy_concerts |   bob   |  analyst
 |    role     |    True   |    True   |      and
 public    | tickit_category_redshift |  table  | policy_concerts |   bob   |  dbadmin
 |    role     |    True   |    True   |      and
```

SVV_RLS_POLICY

使用 SVV_RLS_POLICY 來檢視在 Amazon Redshift 叢集上建立的所有列層級安全政策清單。

所有使用者都可看見 SVV_RLS_POLICY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
poldb	text	列層級安全政策所在資料庫的名稱。

欄名稱	資料類型	描述
polname	text	列層級安全政策的名稱。
polalias	text	政策定義中使用的資料表別名。
polatts	text	提供給政策定義的屬性。
polqual	text	在 CREATE POLICY 陳述式的 USING 子句中提供的政策條件。
polenabled	boolean	政策是否全域開啟。
polmodifiedby	text	建立或最近修改政策的使用者名稱。
polmodifiedtime	timestamp	建立或上次修改政策時的時間戳記。

範例查詢

下列範例顯示 SVV_RLS_POLICY 的結果。

```
-- Create some policies.
CREATE RLS POLICY pol1 WITH (a int) AS t USING ( t.a IS NOT NULL );
CREATE RLS POLICY pol2 WITH (c varchar(10)) AS t USING ( c LIKE '%public%');

-- Inspect the policy in SVV_RLS_POLICY
SELECT * FROM svv_rls_policy;

 poldb | polname | polalias |                    polatts                     |
         polqual                | polenabled | polmodifiedby |   polmodifiedtime
-------+---------+----------+------------------------------------------------
+---------------------------------------+------------+--------------
+---------------------
 my_db | pol1    | t        | [{"colname":"a","type":"integer"}]             |
 "t"."a" IS NOT NULL                   | t          | policy_admin  | 2022-02-11
 14:40:49
 my_db | pol2    | t        | [{"colname":"c","type":"character varying(10)"}] |
 "t"."c" LIKE CAST('%public%' AS TEXT) | t          | policy_admin  | 2022-02-11
 14:41:28
```

SVV_RLS_RELATION

使用 SVV_RLS_RELATION 來檢視受到 RLS 保護的所有關係的清單。

所有使用者都可看見 SVV_RLS_RELATION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
datname	text	包含關聯的資料庫名稱。
relschema	text	包含關係之結構描述的名稱。
relname	text	關係的名稱。
relkind	text	關係的類型，例如資料表或檢視。
is_rls_on	boolean	參數，指出關係是否受到 RS 保護。
is_rls_datashare_on	boolean	此參數指出關係是否透過資料共用受 RLS 保護。
rls_conjunction_type	character(3)	指出關係是否與 and 或 or 結合 RLS 政策的參數。
rls_datashare_conjunction_type	character(3)	指出關係是否透過資料共用與 and 或 or 結合 RLS 政策的參數。

範例查詢

下列範例顯示 SVV_RLS_RELATION 的結果。

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON FOR DATASHARES;


--Inspect RLS state on the relations using SVV_RLS_RELATION.
```

```
SELECT datname, relschema, relname, relkind, is_rls_on, is_rls_datashare_on FROM
 svv_rls_relation ORDER BY relname;

  datname  | relschema |        relname           | relkind | is_rls_on |
 is_rls_datashare_on | rls_conjunction_type | rls_datashare_conjunction_type
----------+-----------+--------------------------+---------+-----------
+---------------------+----------------------+--------------------------------
 tickit_db |   public  | tickit_category_redshift |  table  |      t    |           t
       |          and         |              and
(1 row)
```

SVV_ROLE_GRANTS

使用 SVV_ROLE_GRANTS 來檢視叢集中明確授予角色的角色清單。

下列使用者可以看見 SVV_ROLE_GRANTS：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
role_id	integer	角色的 ID。
role_name	text	角色的名稱。
granted_role_id	integer	授予角色的 ID。
granted_role_name	text	已授予角色的名稱。

範例查詢

下列範例會傳回 SVV_ROLE_GRANTS 的輸出。

```
GRANT ROLE role1 TO ROLE role2;
GRANT ROLE role2 TO ROLE role3;
```

```
SELECT role_name, granted_role_name FROM svv_role_grants;

 role_name |  granted_role_name
-----------+--------------------
   role2   |      role1
   role3   |      role2
(2 rows)
```

SVV_ROLES

使用 SVV_ROLES 來檢視使用者有權存取的角色清單。

下列使用者可以看見 SVV_ROLES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
role_id	integer	規則 ID。
role_name	text	角色的名稱。
role_owner	text	角色擁有者的名稱。
external_id	text	第三方身分提供者中角色的唯一識別碼。

範例查詢

下列範例會傳回 SVV_ROLES 的輸出。

```
SELECT role_name,role_owner FROM svv_roles WHERE role_name IN ('role1', 'role2');

 role_name | role_owner
-----------+------------
```

```
   role1   | superuser
   role2   | superuser
```

SVV_SCHEMA_PRIVILEGES

使用 SVV_SCHEMA_PRIVILEGES 來檢視明確授予目前資料庫中使用者、角色和群組的結構描述許可。

下列使用者可以看見 SVV_SCHEMA_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	指定結構描述所在的命名空間名稱。
privilege_type	text	許可的類型。可能的值是 USAGE 或 CREATE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_SCHEMA_PRIVILEGES 的結果。

```
SELECT namespace_name,privilege_type,identity_name,identity_type,admin_option FROM
 svv_schema_privileges
WHERE namespace_name = 'test_schema1';

 namespace_name | privilege_type |  identity_name | identity_type | admin_option
----------------+----------------+----------------+---------------+--------------
 test_schema1   |    USAGE       |     reguser    |     user      |   False
 test_schema1   |    USAGE       |     role1      |     role      |   False
```

SVV_SCHEMA_QUOTA_STATE

顯示每個結構描述的配額和目前磁碟使用量。

一般使用者可以看到他們對其擁有 USAGE 許可之結構描述的資訊。超級使用者可以看到目前資料庫中所有結構描述的資訊。

所有使用者都可看見 SVV_SCHEMA_QUOTA_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> 只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
schema_id	integer	命名空間或結構描述 ID。
schema_name	character (128)	命名空間或結構描述名稱。
schema_owner	integer	結構描述擁有者的內部使用者 ID。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。

欄名稱	資料類型	描述
disk_usage_pct	double precision	結構描述目前從所設定配額使用的磁碟空間百分比。

範例查詢

下列範例會顯示結構描述的配額和目前磁碟使用量。

```
SELECT TRIM(SCHEMA_NAME) "schema_name", QUOTA, disk_usage, disk_usage_pct FROM
 svv_schema_quota_state
WHERE SCHEMA_NAME = 'sales_schema';
schema_name   | quota | disk_usage | disk_usage_pct
--------------+-------+------------+----------------
sales_schema  | 2048  | 30         | 1.46
(1 row)
```

SVV_SYSTEM_PRIVILEGES

下列使用者可以看見 SVV_SYSTEM_PRIVILEGES：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
system_privilege	text	系統許可的名稱。
identity_id	integer	身分的 ID。可能的值是 user ID 或 role ID。
identity_name	text	身分的名稱。

欄名稱	資料類型	描述
identity_type	text	身分的類型。可能的值是 user 或 role。

範例查詢

下列範例顯示指定參數的結果。

```
SELECT system_privilege,identity_name,identity_type FROM svv_system_privileges
WHERE system_privilege = 'ALTER TABLE' AND identity_name = 'sys:superuser';

 system_privilege | identity_name | identity_type
------------------+---------------+---------------
   ALTER TABLE    | sys:superuser |     role
```

SVV_TABLE_INFO

顯示資料庫中資料表的摘要資訊。檢視會篩選系統資料表並僅顯示使用者定義的資料表。

您可以使用 SVV_TABLE_INFO 檢視來診斷和解決可能影響查詢效能的資料表設計問題。這包括壓縮編碼、分散索引鍵、排序樣式、資料分散扭曲、資料表大小和統計資料的問題。SVV_TABLE_INFO 檢視不會傳回空資料表的任何資訊。

SVV_TABLE_INFO 檢視會從 STV_BLOCKLIST、STV_NODE_STORAGE_CAPACITY、STV_TBL_PERM 和 STV_SLICES 系統資料表和從 PG_DATABASE、PG_ATTRIBUTE、PG_CLASS、PG_NAMESPACE 和 PG_TYPE 目錄資料表中摘要資訊。

只有超級使用者才能看到 SVV_TABLE_INFO。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。若要允許使用者查詢檢視，請將 SVV_TABLE_INFO 上的 SELECT 許可授予給使用者。

資料表欄

欄名稱	資料類型	描述
database	text	資料庫名稱。
schema	text	結構描述名稱。

欄名稱	資料類型	描述
table_id	oid	表格 ID。
table	text	資料表名稱.
encoded	text	值會指出任何欄位是否定義壓縮編碼。
diststyle	text	分散樣式或分散索引鍵欄 (如果鍵分佈有定義的話)。可能的值包括 EVEN、KEY(*column*)、ALL、AUTO(A 、AUTO(EVEN) 及 AUTO(KEY(*column*))。
sortkey1	text	排序索引鍵的第一欄 (如果排序索引鍵有定義的話)。可能的值包括 *column*、AUTO(SORT KEY) 與 AUTO(SORT KEY(*column*))。
max_varchar	integer	使用 VARCHAR 資料類型的最大型欄位大小。
sortkey1_enc	character(32)	排序索引鍵中第一欄的壓縮編碼 (如果排序索引鍵有定義的話)。
sortkey_num	integer	定義為排序索引鍵的欄位編號。
size	bigint	資料表的大小 (以 1-MB 資料區塊表示)。
pct_used	numeric(10,4)	資料表使用的可用空間百分比。

欄名稱	資料類型	描述
empty	bigint	供內部使用。此欄不再使用，並會在未來版本中移除。
unsorted	numeric(5,2)	資料表中未排序之列的百分比。
stats_off	numeric(5,2)	數字會指出資料表統計資訊過時程度。0 為目前，100 為過時。
tbl_rows	numeric(38,0)	資料表中列總數。此值包含標示要進行刪除，但未清空的資料列。
skew_sortkey1	numeric(19,2)	最大非排序索引鍵欄位的大小與排序索引鍵第一欄大小的比率 (如果排序索引鍵有定義的話)。使用此值來評估排序索引鍵是否有效。
skew_rows	numeric(19,2)	資料列數最多的分割，與資料列數最少的分割，兩者之間的資料列數比。
estimated_visible_rows	numeric(38,0)	資料表中的估計資料列，此值不包含標記刪除的資料列。

欄名稱	資料類型	描述

risk_event	text	資料表的相關風險資訊。該欄位會分成以下幾個部分：
		
		`risk_type |xid|timestamp`
		
		• risk_type ，其中 1 表示 COPY command with the EXPLICIT_IDS option 已執行。Amazon Redshift 不再檢查資料表中 IDENTITY 欄的唯一性。如需詳細資訊，請參閱 EXPLICIT_IDS。
		• xid 交易 ID 會說明風險。
		• timestamp 表示 COPY 命令的執行時間。
		
		下列範例顯示欄位中的值。
		
		`1|1107|2019-06-22 07:16:11.292952`

vacuum_sort_benefit	numeric(12,2)	在您執行清空排序時掃描查詢效能的估計改善百分比上限。
create_time	沒有時區的時間戳記	建立資料表時的時間戳記。

範例查詢

下列範例顯示資料庫中所有使用者定義資料表的編碼、分佈樣式、排序和資料扭曲。此處的 "table" 必須使用雙引號括起來，因為這是保留字。

```
select "table", encoded, diststyle, sortkey1, skew_sortkey1, skew_rows
```

```
from svv_table_info
order by 1;

table          | encoded | diststyle       | sortkey1     | skew_sortkey1 | skew_rows
---------------+---------+-----------------+--------------+---------------+----------
category       | N       | EVEN            |              |               |
date           | N       | ALL             | dateid       |          1.00 |
event          | Y       | KEY(eventid)    | dateid       |          1.00 |      1.02
listing        | Y       | KEY(listid)     | dateid       |          1.00 |      1.01
sales          | Y       | KEY(listid)     | dateid       |          1.00 |      1.02
users          | Y       | KEY(userid)     | userid       |          1.00 |      1.01
venue          | N       | ALL             | venueid      |          1.00 |
(7 rows)
```

SVV_TABLES

使用 SVV_TABLES 來檢視本機和外部目錄中的資料表。

所有使用者都可看見 SVV_TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
table_catalog	text	資料表所在之目錄的名稱。
table_schema	text	資料表的結構描述名稱。
table_name	text	表格的名稱。
table_type	text	資料表的類型。可能的值是 views、external tables 和 base tables。
備註	text	備註。

SVV_TRANSACTIONS

記錄資料庫中目前保持鎖定之資料表的交易相關資訊。使用 SVV_TRANSACTIONS 檢視以識別開啟交易與鎖定爭用的問題。如需鎖定的相關資訊，請參閱 管理並行寫入操作 和 LOCK。

所有使用者都可看見 SVV_TRANSACTIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
txn_owner	text	交易之擁有者的名稱。
txn_db	text	與交易相關聯之資料庫的名稱。
xid	bigint	交易 ID。
pid	integer	與鎖定關聯的處理程序 ID。
txn_start	timestamp	交易的開始時間。
lock_mode	text	此程序保留或請求的鎖定模式名稱。如果 `lock_mode` 是 `ExclusiveLock` 且 `granted` 為 true (t)，則此交易 ID 是開放交易。
lockable_object_type	text	請求或保留鎖定的物件類型，`relation` (如果是資料表) 或 `transactionid` (如果是交易)。
關聯	integer	取得鎖定之資料表 (關聯) 的資料表 ID。此值是 NULL (如果 `lockable_object_type` 是 `transactionid`)。

欄名稱	資料類型	描述
獲授予	boolean	值會指出鎖定是否獲授予 (t) 或待定 (f)。

範例查詢

下列命令顯示所有作用中的交易且每個交易請求的鎖定數。

```
select * from svv_transactions;

 txn_
 lockable_
 owner | txn_db |  xid   |  pid  |         txn_start          |      lock_mode      |
 object_type    | relation | granted
-------+--------+--------+-------+----------------------------+---------------------
+----------------+----------+---------
 root  | dev    | 438484 | 22223 | 2016-03-02 18:42:18.862254 | AccessShareLock     |
 relation       |   100068 | t
 root  | dev    | 438484 | 22223 | 2016-03-02 18:42:18.862254 | ExclusiveLock       |
 transactionid  |          | t
 root  | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | AccessShareLock     |
 relation       |    50860 | t
 root  | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | AccessShareLock     |
 relation       |    52310 | t
 root  | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | ExclusiveLock       |
 transactionid  |          | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100068 | f
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | RowExclusiveLock    |
 relation       |    16688 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessShareLock     |
 relation       |   100064 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100166 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100171 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
 relation       |   100190 | t
 root  | dev    | 438505 | 22378 | 2016-03-02 18:43:27.611292 | ExclusiveLock       |
 transactionid  |          | t
(12 rows)
```

```
(12 rows)
```

SVV_USER_GRANTS

使用 SVV_USER_GRANTS 來檢視叢集中明確授予角色的使用者清單。

下列使用者可以看見 SVV_USER_GRANTS：

- 超級使用者
- 具有 ACCESS SYSTEM TABLE 許可的使用者

其他使用者只能看到明確授予他們的角色。

資料表欄

欄名稱	資料類型	描述
user_id	integer	使用者的使用者 ID。
user_name	text	使用者的名稱。
role_id	integer	授予角色的角色 ID。
role_name	text	授予角色的角色名稱。
admin_option	boolean	指出使用者是否可以將角色授予其他使用者和角色的值。

範例查詢

下列查詢會將角色授予使用者，並顯示明確授予角色的使用者清單。

```
GRANT ROLE role1 TO reguser;
GRANT ROLE role2 TO reguser;
GRANT ROLE role1 TO superuser;
GRANT ROLE role2 TO superuser;

SELECT user_name,role_name,admin_option FROM svv_user_grants;
```

```
 user_name | role_name | admin_option
-----------+-----------+--------------
 superuser |  role1    | False
 reguser   |  role1    | False
 superuser |  role2    | False
  reguser  |  role2    | False
```

SVV_USER_INFO

您可以使用 SVV_USER_INFO 檢視來擷取關於 Amazon Redshift 資料庫使用者的資料。

所有使用者都可看見 SVV_USER_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_name	text	角色的使用者名稱。
user_id	integer	使用者的使用者 ID。
createdb	boolean	一個值，指出使用者是否具有建立資料庫的許可。
superuser	boolean	一個值，指出使用者是否為超級使用者。
catalog_update	boolean	一個值，指出使用者可以更新系統目錄。
connection_limit	text	使用者可以開啟的連線數量。
syslog_access	text	一個值，指出使用者是否具有系統日誌的存取權。兩個可能值為 RESTRICTED 和 UNRESTRICTED 。RESTRICTED 表示不是超級使用者的使用者可以看到其記錄。UNRESTRICTED 表示不是超級使用者的使用者可以看到他們具有 SELECT 權限的系統檢視和資料表中的所有記錄。
last_ddl_timestamp	timestamp	使用者執行的最新資料定義語言 (DDL) create 陳述式的時間戳記。

欄名稱	資料類型	描述
session_timeout	integer	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。如需叢集的閒置或非作用中逾時設定詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制。
external_user_id	text	第三方身分提供者中使用者的唯一識別碼。

範例查詢

下列命令會從 SVV_USER_INFO 擷取使用者資訊。

```
SELECT * FROM SVV_USER_INFO;
```

SVV_VACUUM_PROGRESS

此檢視會傳回目前進行中的清空操作所需完成時間的預估值。

只有超級使用者才能看到 SVV_VACUUM_PROGRESS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_VACUUM_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需 SVV_VACUUM_SUMMARY 的相關資訊，請參閱 SVV_VACUUM_SUMMARY。

如需 SVL_VACUUM_PERCENTAGE 的相關資訊，請參閱 SVL_VACUUM_PERCENTAGE。

> **Note**
>
> 只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
table_name	text	目前正受到清空或前次受到清空的資料表名稱 (如果沒有任何操作在進行中)。
status	text	描述在清空操作過程中所做的目前活動： • 初始化 • Sort • Merge • Delete • Select • 失敗 • 完成 • 略過 • 建置 INTERLEAVED SORTKEY 順序
time_remaining_estimate	text	目前清空操作所剩需完成的預估時間 (以分和秒表示)：例如，**5m 10s**。在清空完成其第一次的排序操作前都不會將預估時間傳回。如果沒有任何進行中的清空，則會顯示前次執行的清空，並在 STATUS 欄中顯示 **Completed** ，且 TIME_REMAINING_ESTIMATE 欄為空。預估值隨著清空的進行通常會變得越來越精準。

範例查詢

幾分鐘前執行的下列查詢會顯示正在清空之名為 SALESNEW 的大型資料表。

```
select * from svv_vacuum_progress;

table_name    |            status              | time_remaining_estimate
--------------+--------------------------------+-------------------------
salesnew      |  Vacuum: initialize salesnew  |
```

```
(1 row)
...
select * from svv_vacuum_progress;

table_name   |         status         | time_remaining_estimate
-------------+------------------------+-------------------------
salesnew     |  Vacuum salesnew sort  | 33m 21s
(1 row)
```

下列查詢顯示目前沒有在進行任何清空操作。前次執行清空的資料表是 SALES 資料表。

```
select * from svv_vacuum_progress;

table_name   |  status  | time_remaining_estimate
-------------+----------+-------------------------
  sales      | Complete |
(1 row)
```

SVV_VACUUM_SUMMARY

SVV_VACUUM_SUMMARY 檢視會結合 STL_VACUUM、STL_QUERY、STV_TBL_PERM 資料表，摘要系統所記錄之清空操作的相關資訊。檢視會根據清空交易和資料表傳回一個資料列。檢視會記錄操作的經過時間、建立之排序分割區數、所需合併增量數和操作執行前後的資料列和區塊計數的差異。

只有超級使用者才能看到 SVV_VACUUM_SUMMARY。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_VACUUM_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需 SVV_VACUUM_PROGRESS 的相關資訊，請參閱 SVV_VACUUM_PROGRESS。

如需 SVL_VACUUM_PERCENTAGE 的相關資訊，請參閱 SVL_VACUUM_PERCENTAGE。

> **Note**
>
> 只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
table_name	text	清空的資料表名稱。
xid	bigint	VACUUM 操作的交易 ID。
sort_partitions	bigint	在清空操作的排序階段所建立之排序分割區數。
merge_increments	bigint	合併增量數，需要此項目來完成清空操作的合併階段。
elapsed_time	bigint	清空操作的經過執行期 (微秒)。
row_delta	bigint	清空前後資料表資料列數總計的差異。
sortedrow_delta	bigint	清空前後排序資料表資料列數的差異。
block_delta	integer	清空前後資料表區塊計數的差異。
max_merge_partitions	integer	會使用此欄位來分析表現並表示分割區數上限，也就是清空可以根據合併階段反覆運算為資料表處理的分割區數上限。(真空將未分類的區域排序為一個或多個已排序的分區。根據資料表中的欄數和目前的 Amazon Redshift 組態，合併階段可以在單次合併反覆運算中處理最大數量的分割區。如果已排序的分割區數目超過合併分割區的最大數目，合併階段仍然可以運作，但需要更多的合併反覆運算。)

範例查詢

下列查詢會傳回在三個不同資料表上清空操作的統計資訊。SALES 資料表執行了兩次清空。

```
select table_name, xid, sort_partitions as parts, merge_increments as merges,
elapsed_time, row_delta, sortedrow_delta as sorted_delta, block_delta
from svv_vacuum_summary
order by xid;

table_  | xid  |parts|merges| elapsed_ | row_    | sorted_ | block_
```

```
name     |      |     |      | time     | delta    | delta    | delta
---------+------+-----+------+----------+----------+----------+--------
users    | 2985 |   1 |    1 | 61919653 |        0 |    49990 |      20
category| 3982 |   1 |    1 | 24136484 |        0 |       11 |       0
sales    | 3992 |   2 |    1 | 71736163 |        0 | 1207192 |      32
sales    | 4000 |   1 |    1 | 15363010 | -851648 | -851648 |    -140
(4 rows)
```

SYS 監控檢視

監控檢視是 Amazon Redshift 中的系統檢視，用於監控佈建叢集和無伺服器工作群組的查詢和工作負載資源使用情況。這些檢視位於 pg_catalog 結構描述中。若要顯示這些檢視提供的資訊，請執行 SQL SELECT 陳述式。

除非另有說明，否則這些檢視可用於 Amazon Redshift 叢集和 Amazon Redshift Serverless 工作群組。

SYS_SERVERLESS_USAGE 只收集 Amazon Redshift Serverless 的使用資料。

主題

- SYS_LOAD_DETAIL
- SYS_LOAD_ERROR_DETAIL
- SYS_LOAD_HISTORY
- SYS_MV_REFRESH_HISTORY
- SYS_MV_STATE
- SYS_PROCEDURE_CALL
- SYS_PROCEDURE_MESSAGES
- SYS_QUERY_DETAIL
- SYS_QUERY_HISTORY
- SYS_QUERY_TEXT
- SYS_RESTORE_LOG
- SYS_RESTORE_STATE
- SYS_SCHEMA_QUOTA_VIOLATIONS
- SYS_SERVERLESS_USAGE
- SYS_SESSION_HISTORY
- SYS_SPATIAL_SIMPLIFY
- SYS_STREAM_SCAN_ERRORS
- SYS_STREAM_SCAN_STATES
- SYS_TRANSACTION_HISTORY
- SYS_UDF_LOG
- SYS_UNLOAD_DETAIL
- SYS_UNLOAD_HISTORY
- SYS_USERLOG
- SYS_VACUUM_HISTORY

SYS_ANALYZE_COMPRESSION_HISTORY

記錄在 COPY 或 ANALYZE COMPRESSION 命令期間的壓縮分析操作詳細資訊。

所有使用者都可看見 SYS_ANALYZE_COMPRESSION_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
start_time	timestamp	壓縮分析操作開始進行的時間。
transaction_id	bigint	壓縮分析操作的交易 ID。
table_id	integer	接受分析之資料表的 ID。
table_name	character(128)	接受分析之資料表的名稱。
column_position	integer	接受分析之資料表的資料欄索引，用於判斷壓縮編碼。
old_encoding	character(15)	壓縮分析之後的編碼類型。
new_encoding	character(15)	壓縮分析之後的編碼類型。
模式	character(14)	可能值如下： PRESET 指定 new_encoding 是由 Amazon Redshift COPY 命令根據欄資料類型決定。不會對任何資料取樣。 ON 指定 new_encoding 是由 Amazon Redshift COPY 命令根據範例資料的分析決定。 ANALYZE ONLY 指定 new_encoding 是由 Amazon Redshift ANALYZE COMPRESSION 命令根據範例資料的分析決定。然而，不會變更已分析資料欄的編碼類型。

範例查詢

下列範例由在相同工作階段中執行的最後一項 COPY 命令檢查 lineitem 資料表上的壓縮分析詳細資料。

```
select transaction_id, table_id, btrim(table_name) as table_name, column_position,
 old_encoding, new_encoding, mode
from sys_analyze_compression_history
where transaction_id = (select transaction_id from sys_query_history where query_id =
 pg_last_copy_id()) order by column_position;

 transaction_id  |  table_id   | table_name | column_position |  old_encoding   |
 new_encoding   |      mode
-----------------+-------------+------------+-----------------+-----------------
+-----------------+-------------
      8196       |   248126    | lineitem   |        0        | mostly32        |
 mostly32        | ON
      8196       |   248126    | lineitem   |        1        | mostly32        | lzo
           | ON
      8196       |   248126    | lineitem   |        2        | lzo             |
 delta32k        | ON
      8196       |   248126    | lineitem   |        3        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |        4        | bytedict        |
 bytedict        | ON
      8196       |   248126    | lineitem   |        5        | mostly32        |
 mostly32        | ON
      8196       |   248126    | lineitem   |        6        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |        7        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |        8        | lzo             | zstd
           | ON
      8196       |   248126    | lineitem   |        9        | runlength       | zstd
           | ON
      8196       |   248126    | lineitem   |       10        | delta           | lzo
           | ON
      8196       |   248126    | lineitem   |       11        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |       12        | delta           | delta
           | ON
      8196       |   248126    | lineitem   |       13        | bytedict        | zstd
           | ON
```

```
      8196       |   248126    | lineitem   |       14        | bytedict        | zstd
           | ON
      8196       |   248126    | lineitem   |       15        | text255         | zstd
           | ON
(16 rows)
```

SYS_ANALYZE_HISTORY

記錄 ANALYZE 操作的詳細資訊。

只有超級使用者才能看到 SYS_ANALYZE_HISTORY。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
transaction_id	long	交易的 ID。
query_id	long	SYS_QUERY_HISTORY 中的查詢識別碼。
database_name	char(30)	資料庫的名稱。
table_name	char(30)	資料表的名稱。
table_id	integer	資料表的 ID。
is_automatic	char(1)	如果操作預設包含 Amazon Redshift ANALYZE 操作，則值為 true (t)。如果明確執行 ANALYZE 命令，則值為 false (f)。
status	char(15)	分析命令的結果。可能的值為「完整」、「略過」和 Predicate Column。
start_time	timestamp	ANALYZE 操作開始執行的時間，以 UTC 表示。
end_time	timestamp	ANALYZE 操作完成執行的時間，以 UTC 表示。

欄名稱	資料類型	描述
rows	double	資料表中的列總數
modified_rows	double	自前次 ANALYZE 操作後已修改的資料列總數。
analyze_threshold_percent	integer	analyze_threshold_percent 參數的值。
last_analyze_time	timestamp	先前分析資料表的時間，以 UTC 表示。

範例查詢

```
 user_id | transaction_id | database_name | schema_name |      table_name     |
 table_id | is_automatic | Status |         start_time         |          end_time
     | rows | modified_rows | analyze_threshold_percent |  last_analyze_time
---------+----------------+---------------+-------------+---------------------
+----------+--------------+--------+----------------------------+----------
+-----------------+------+--------------+--------------------------
+---------------------
     101 |           8006 |           dev |      public | test_table_562bf8dc
 |   110427 |            f |   Full | 2023-09-21 18:33:08.504646 | 2023-09-21
 18:33:24.296498 |    5 |             5 |                          0 | 2000-01-01
 00:00:00
```

系統應用程序掩碼策略日誌

使用 SYS_APPLICKING_LOG 來追蹤參考受 DDM 保護之關係之查詢的動態資料遮罩原則的應用程式。

下列使用者可以看見系統應用程式遮罩 _ 政策 _ 日誌：

- 超級使用者
- 具有 sys:operator 角色的使用者

- 具有 ACCESS SYSTEM TABLE 許可的使用者

一般使用者將看到 0 列。

請注意，具有該角色的使用者看不到 SYS_APPLICK。`sys:secadmin`

如需動態資料遮罩的詳細資訊，請移至動態資料遮罩。

資料表欄

欄名稱	資料類型	描述
policy_name	text	遮罩政策的名稱。
user_id	text	執行查詢的使用者識別碼。
record_time	timestamp	記錄系統檢視項目的時間。
session_id	int	程序 ID。
transaction_id	long	交易的 ID。
query_id	int	查詢 ID。
database_name	text	執行查詢的資料庫名稱。
relation_name	text	套用遮罩原則的表格名稱。
schema_name	text	資料表所在的結構描述名稱。
附件識別碼	long	附加的遮罩原則識別碼。
關係種類	text	套用遮罩原則的關係類型。可能值為 TABLE、VIEW、LATE BINDING VIEW 和 MATERIALIZED VIEW 。

範例查詢

下列範例顯示mask_credit_card_full遮罩原則已附加至credit_db.public.credit_cards表格。

```
select policy_name, database_name, relation_name, schema_name, relation_kind
from sys_applied_masking_policy_log;

policy_name           | database_name | relation_name | schema_name | relation_kind
----------------------+---------------+---------------+-------------+--------------
mask_credit_card_full | credit_db     | credit_cards  | public      | table

(1 row)
```

系統自動表優化

記錄 Amazon Redshift 對自動最佳化而定義的資料表執行的自動操作。

只有超級使用者才能看到 SYS_自動表格最佳化。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
transaction_id	long	交易識別碼。
session_id	int	執行 alter 命令的進程的會話標識符。
table_id	int	資料表識別碼。
替代表格類型	character (32)	建議的類型。可能值為 distkey、sortkey 及 encode。
status	character (128)	建議的完成狀態。可能值為 Start、Complete、Skipped、Abort、Checkpoint 以及 Failed。

欄名稱	資料類型	描述
event_time	timestamp	狀態欄的時間戳記。
替代 (_ 從)	character (200)	套用建議之前，資料表的先前分散樣式和排序索引鍵。該值會被截斷為 200 個字元的增量。
替代至	character (200)	套用建議之後，表格的目前分佈樣式和排序索引鍵。該值會被截斷為 200 個字元的增量。

範例查詢

在下列範例中，結果中的列顯示 Amazon Redshift 所採取的動作。

```
SELECT table_id, alter_table_type, status, event_time, alter_from
FROM SYS_AUTO_TABLE_OPTIMIZATION;

 table_id |  alter_table_type   |                        status
 |         event_time          |  alter_from
----------+---------------------+-----------------------------------------------------
+-----------------------------+-----------------
   118082 | sortkey             | Start
 | 2020-08-22 19:42:20.727049  |
   118078 | sortkey             | Start
 | 2020-08-22 19:43:54.728819  |
   118082 | sortkey             | Start
 | 2020-08-22 19:42:52.690264  |
   118072 | sortkey             | Start
 | 2020-08-22 19:44:14.793572  |
   118082 | sortkey             | Failed
 | 2020-08-22 19:42:20.728917  |
   118078 | sortkey             | Complete
 | 2020-08-22 19:43:54.792705  |  SORTKEY: None;
   118086 | sortkey             | Complete
 | 2020-08-22 19:42:00.72635   |  SORTKEY: None;
   118082 | sortkey             | Complete
 | 2020-08-22 19:43:34.728144  |  SORTKEY: None;
   118072 | sortkey             | Skipped:Retry exceeds the maximum limit for a table.
 | 2020-08-22 19:44:46.706155  |
```

```
  118086 | sortkey             | Start
| 2020-08-22 19:42:00.685255  |
  118082 | sortkey             | Start
| 2020-08-22 19:43:34.69531   |
  118072 | sortkey             | Start
| 2020-08-22 19:44:46.703331  |
  118082 | sortkey             | Checkpoint: progress 14.755079%
| 2020-08-22 19:42:52.692828  |
  118072 | sortkey             | Failed
| 2020-08-22 19:44:14.796071  |
  116723 | sortkey             | Abort:This table is not AUTO.
| 2020-10-28 05:12:58.479233  |
  110203 | distkey             | Abort:This table is not AUTO.
| 2020-10-28 05:45:54.67259   |
```

SYS_CONNECTION_LOG

記錄身分驗證嘗試以及連線和中斷連線。

只有超級使用者才能看到 SYS_CONNECTION_LOG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
事件	character(50)	連線或身分驗證事件。
record_time	timestamp	發生事件的時間。
remote_host	character(45)	遠端主機的名稱或 IP 地址。
remote_port	character(32)	遠端主機的連接埠號碼。
session_id	integer	與陳述式相關聯的處理程序 ID。
database_name	character(50)	資料庫名稱。
user_name	character(50)	使用者名稱。

欄名稱	資料類型	描述
auth_method	character(32)	身分驗證方法。
持續時間	integer	連線的持續時間，以微秒為單位。
ssl_version	character(50)	Secure Sockets Layer (SSL) 版本。
ssl_cipher	character(128)	SSL 密碼。
mtu	integer	最大傳輸單位 (MTU)。
ssl_compression	character(64)	SSL 壓縮類型。
ssl_expansion	character(64)	SSL 擴展類型。
iam_auth_guid	character(36)	請求的 IAM 身份驗證 CloudTrail ID。
application_name	character(250)	工作階段之應用程式的初始或已更新名稱。
driver_version	character(64)	從第三方 SQL 用戶端工具連線到 Amazon Redshift 叢集的 ODBC 或 JDBC 驅動程式版本。
os_version	character(64)	連線到 Amazon Redshift 叢集之用戶端機器上的作業系統版本。
plugin_name	character(32)	連接至您的 Amazon Redshift 叢集時使用的外掛程式名稱。

欄名稱	資料類型	描述
protocol_version	integer	Amazon Redshift 驅動程式在建立與伺服器的連線時使用的內部通訊協定版本。通訊協定版本會在驅動程式與伺服器之間進行交涉。該版本描述了可用的功能。有效值包含： • 0 (BASE_SERVER_PROTOCOL_VERSION) • 1 (EXTENDED_RESULT_METADATA_SERVER_PROTOCOL_VERSION) - 為了儲存每個查詢的往返，伺服器會傳送額外的結果集中繼資料資訊。 • 2 (BINARY_PROTECOL_VERSION) - 根據結果集的資料類型，伺服器會以二進位格式傳送資料。 • 3 (EXTENDED2_RESULT_METADATA_SERVER_PROTOCOL_VERSION) – 伺服器傳送列的區分大小寫 (排序規則) 資訊。
global_session_id	character(36)	目前工作階段的全域唯一識別碼。工作階段 ID 在節點故障重新啟動後仍然存在。

範例查詢

若要檢視已開啟之連線的詳細資訊，請執行下列查詢。

```
select record_time, user_name, database_name, remote_host, remote_port
from sys_connection_log
where event = 'initiating session'
and session_id not in
(select session_id from sys_connection_log
where event = 'disconnecting session')
order by 1 desc;

record_time         | user_name   | database_name   | remote_host   | remote_port

--------------------+-------------+-----------------+---------------
+---------------------------------
2014-11-06 20:30:06 | rdsdb       | dev             | [local]       |

2014-11-06 20:29:37 | test001     | test            | 10.49.42.138  | 11111
```

```
2014-11-05 20:30:29 | rdsdb       | dev             | 10.49.42.138  | 33333

2014-11-05 20:28:35 | rdsdb       | dev             | [local]       |
(4 rows)
```

下列範例反映失敗的身分驗證嘗試，以及成功的連線和中斷連線。

```
select event, record_time, remote_host, user_name
from sys_connection_log order by record_time;

            event      |         record_time        |  remote_host  | user_name

-----------------------+----------------------------+---------------+---------
authentication failure | 2012-10-25 14:41:56.96391  | 10.49.42.138  | john

authenticated          | 2012-10-25 14:42:10.87613  | 10.49.42.138  | john

initiating session     | 2012-10-25 14:42:10.87638  | 10.49.42.138  | john

disconnecting session  | 2012-10-25 14:42:19.95992  | 10.49.42.138  | john

(4 rows)
```

下列範例顯示 ODBC 驅動程式的版本、用戶端機器上的作業系統，以及用來連線到 Amazon Redshift 叢集的外掛程式。在此範例中，使用的外掛程式用於使用登入名稱和密碼進行標準 ODBC 驅動程式驗證。

```
select driver_version, os_version, plugin_name from sys_connection_log;

driver_version                          |  os_version                       |
 plugin_name
-----------------------------------------+-----------------------------------
+--------------------
Amazon Redshift ODBC Driver 1.4.15.0001 | Darwin 18.7.0 x86_64              | none
Amazon Redshift ODBC Driver 1.4.15.0001 | Linux 4.15.0-101-generic x86_64   | none
```

下列範例顯示用戶端電腦上的作業系統版本、驅動程式版本和通訊協定版本。

```
select os_version, driver_version, protocol_version from sys_connection_log;

os_version                      |  driver_version             | protocol_version
```

```
--------------------------------+------------------------------+--------------------
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
```

SYS_COPY_JOB (預覽)

這是預覽版本中的自動複製 (SQL 複製工作) 的發行前文件。文件和功能會隨時變更。我們建議僅在測試環境中使用此功能，不要在生產環境中使用。公開預覽將於 2024 年 7 月 31 日結束。預覽叢集將在預覽版結束的兩週後自動移除。如需預覽版條款和條件，請參閱 AWS 服務條款中的 Beta 版和預覽版。

使用 SYS_COPY_JOB 來檢視 COPY JOB 命令的詳細資訊。

此檢視包含已建立的 COPY JOB 命令。

所有使用者都可看見 SYS_COPY_JOB。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
job_id	bigint	複製任務識別碼。
job_name	character(128)	複製任務的名稱。
iam_role	character(128)	在 COPY 陳述式中指定的 IAM 角色。
job_text	character(256)	COPY 陳述式的參數。
is_auto	integer	指出 COPY JOB 是否由 Amazon Redshift 自動執行。1 表示 true，0 表示 false。
on_error_suspend	integer	此資訊僅供內部使用。

SYS_COPY_REPLACEMENTS

顯示一個日誌，其記錄搭配 ACCEPTINVCHARS 選項的 COPY 命令何時取代無效的 UTF-8 字元。在至少需要一個取代項目的每個節點上，對於其前 100 個列的每一個都會新增一個日誌項目至 SYS_COPY_REPLACEMENTS。

您可以使用此檢視來查看有關無伺服器工作群組和已佈建叢集的資訊。

所有使用者都可看見 SYS_COPY_REPLACEMENTS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生查詢的使用者之 ID。
query_id	bigint	查詢 ID。用於聯結其他系統資料表與檢視的欄。
table_id	integer	資料表 ID。
file_name	character (256)	用於 COPY 命令之輸入檔案的完整路徑。
column_name	character (127)	包含無效 UTF-8 字元的第一個欄位。
line_number	bigint	輸入資料檔中包含無效 UTF-8 字元的行號。-1 表示無法使用行號，例如從單欄式資料檔案複製時。
raw_line	character (1024)	包含無效 UTF-8 字元的原始載入資料。

範例查詢

下列範例會傳回最新 COPY 操作的取代項目。

```
select query_idp, table_id, file_name, line_number, colname
from sys_copy_replacements
```

```
where query = pg_last_copy_id();

 query_id | table_id |   file_name                          | line_number | column_name
 ---------+----------+--------------------------------------+-------------+--------
    96    |    26    | s3://mybucket/allusers_pipe.txt      |         123 | city
    96    |    26    | s3://mybucket/allusers_pipe.txt      |         456 | city
    96    |    26    | s3://mybucket/allusers_pipe.txt      |         789 | city
    96    |    26    | s3://mybucket/allusers_pipe.txt      |         012 | city
    96    |    26    | s3://mybucket/allusers_pipe.txt      |         119 | city
...
```

SYS_DATASHARE_CHANGE_LOG

記錄用於追蹤生產者和消費者叢集上資料共用變更的合併檢視。

所有使用者都可看見 SYS_DATASHARE_CHANGE_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	採取動作之使用者的 ID。
user_name	varchar(128)	採取動作之使用者的名稱。
session_id	integer	工作階段的 ID。
transaction_id	bigint	交易的 ID。
share_id	integer	受影響資料共用的 ID。
share_name	varchar(128)	資料共用的名稱。
source_database_id	integer	資料共用所屬資料庫的 ID。

欄名稱	資料類型	描述
source_database_name	varchar(128)	資料共用所屬資料庫的名稱。
consumer_database_id	integer	從資料共用匯入之資料庫的 ID。
consumer_database_name	varchar(128)	從資料共用匯入之資料庫的名稱。
arn	varchar(192)	支援匯入資料庫之資源的 ARN。
record_time	timestamp	動作的時間戳記。
動作	varchar(128)	正在執行的動作。可能的值包括創建數據保護，刪除數據保護，授予更改，撤銷更改，授予共享，撤銷份額，更改添加，更改刪除，更改設置，授予使用情況，撤銷使用情況，創建數據庫，授予或撤銷共享數據庫上的使用情況，刪除共享數據庫，改變共享數據庫。
status	integer	動作的狀態。可能的值為 SUCCESS 和 ERROR-ERROR CODE。
share_object_type	varchar(64)	新增至資料共用或從資料共用中移除之資料庫物件的類型。可能的值包括 schemas、tables、columns、functions 和 views。這是生產者叢集的欄位。
share_object_id	integer	新增至資料共用或從資料共用中移除之資料庫物件的 ID。這是生產者叢集的欄位。
share_object_name	varchar(128)	新增至資料共用或從資料共用中移除之資料庫物件的名稱。這是生產者叢集的欄位。
target_user_type	varchar(16)	授與權限的使用者或群組類型。這是生產者和消費者叢集的欄位。

欄名稱	資料類型	描述
target_user_id	integer	授與權限的使用者或群組 ID。這是生產者和消費者叢集的欄位。
target_user_name	varchar(128)	授與權限的使用者或群組名稱。這是生產者和消費者叢集的欄位。
consumer_account	varchar(16)	資料消費者的帳戶 ID。這是生產者叢集的欄位。
consumer_namespace	varchar(64)	資料消費者帳戶的命名空間。這是生產者叢集的欄位。
producer_account	varchar(16)	資料共用所屬生產者帳戶的帳戶 ID。這是消費者叢集的欄位。
producer_namespace	varchar(64)	資料共用所屬產品帳戶的命名空間。這是消費者叢集的欄位。
attribute_name	varchar(64)	資料共用或共用資料庫的屬性名稱。
attribute_value	varchar(128)	資料共用或共用資料庫的屬性值。
message	varchar(512)	動作失敗時的錯誤訊息。

範例查詢

下列範例顯示 SYS_DATASHARE_CHANGE_LOG 檢視。

```
SELECT DISTINCT action
FROM sys_datashare_change_log
WHERE share_object_name LIKE 'tickit%';

         action
 ----------------------
```

```
  "ALTER DATASHARE ADD"
```

SYS_DATASHARE_CROSS_REGION_USAGE

使用 SYS_DATASHARE_CROSS_REGION_USAGE 檢視，可取得跨區域資料共用查詢所造成之跨區域資料傳輸使用量的摘要。SYS_DATASHARE_CROSS_REGION_USAGE 彙總區段層級的詳細資料。

只有超級使用者才能看到 SYS_DATASHARE_CROSS_REGION_USAGE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
query_id	integer	查詢的 ID。使用此值來聯結各種其他系統資料表與檢視。
child_query_sequence	integer	重寫使用者查詢的順序，從 1 開始。
segment_id	bigint	區段的號碼。查詢包含多個區段，每個區段包含一或多個步驟。
start_time	time	資料傳輸開始的時間 (以 UTC 為單位)。
end_time	time	資料傳輸結束的時間 (以 UTC 為單位)。
transferred_data	bigint	從生產者區域傳輸至消費者區域的資料位元組數量。
source_region	char(25)	查詢傳輸資料的來源生產者區域。

範例查詢

下列範例顯示 SYS_DATASHARE_CROSS_REGION_USAGE 檢視。

```
SELECT query, segment, transferred_data, source_region
```

```
from sys_datashare_cross_region_usage
where query = pg_last_query_id()
order by query,segment;

  query | segment | transferred_data | source_region
--------+---------+------------------+---------------
 200048 |       2 |          4194304 |    us-west-1
 200048 |       2 |          4194304 |    us-east-2
```

SYS_DATASHARE_USAGE_CONSUMER

記錄資料共用的活動和使用情況。此檢視僅與消費者叢集相關。

所有使用者都可看見 SYS_DATASHARE_USAGE_CONSUMER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	發出要求的使用者 ID。
session_id	integer	執行查詢之領導者程序的 ID。
transaction_id	bigint	目前交易的內容。
request_id	varchar(50)	要求之 API 呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。
transaction_uid	varchar(50)	交易的唯一 ID。
record_time	timestamp	記錄動作的時間。
status	integer	要求的 API 呼叫的狀態。

欄名稱	資料類型	描述
error_message	varchar(512)	錯誤的訊息。

範例查詢

下列範例顯示 SYS_DATASHARE_USAGE_CONSUMER 檢視。

```
SELECT request_type, status, trim(error) AS error
FROM sys_datashare_usage_consumer

  request_type  | status | error_message
----------------+--------+---------------
 "GET RELATION" |   0    |
```

SYS_DATASHARE_USAGE_PRODUCER

記錄資料共用的活動和使用情況。此檢視僅與生產者叢集相關。

所有使用者都可看見 SYS_DATASHARE_USAGE_PRODUCER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
share_id	integer	資料共用的物件 ID (OID)。
share_name	varchar(128)	資料共用的名稱。
request_id	varchar(50)	要求之 API 呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。

欄名稱	資料類型	描述
object_type	varchar(64)	從資料共用共用的物件類型。可能的值包括 schemas、tables、columns、functions 和 views。
object_oid	integer	從資料共用共用的物件 ID。
object_name	varchar(128)	要從資料共用共用的物件名稱。
consumer_account	varchar(16)	資料共用共用到的消費者帳戶的帳戶。
consumer_namespace	varchar(64)	資料共用共用到的使用者帳戶的命名空間。
consumer_transaction_uid	varchar(50)	消費者叢集上陳述式的唯一交易 ID。
record_time	timestamp	記錄動作的時間。
status	integer	資料共用的狀態。
error_message	varchar(512)	錯誤的訊息。
consumer_region	char(64)	取用者叢集所在的區域。

範例查詢

下列範例顯示 SYS_DATASHARE_USAGE_PRODUCER 檢視。

```
SELECT DISTINCT
FROM sys_datashare_usage_producer
WHERE object_name LIKE 'tickit%';

   request_type
```

```
------------------
  "GET RELATION"
```

SYS_EXTERNAL_QUERY_DETAIL

使用 SYS_EXTERNAL_QUERY_DETAIL 來檢視區段層級的查詢詳細資料。每一列代表特定 WLM 查詢的區段，其中包含處理的列數、處理的位元組數以及 Amazon S3 中外部資料表的分割區資訊等詳細資訊。此檢視中的每一列在 SYS_QUERY_DETAIL 檢視中也會有一個對應的項目，但此檢視包含與外部查詢處理相關的更多詳細資訊。

所有使用者都可看見 SYS_EXTERNAL_QUERY_DETAIL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	外部查詢的查詢識別碼。
transaction_id	bigint	交易識別碼。
child_query_sequence	integer	重寫使用者查詢的順序。從 0 開始，類似於 segment_id。
segment_id	integer	查詢區段的區段識別碼。
source_type	character(32)	查詢的資料來源類型，對於 Redshift Spectrum 可以是 S3，對於聯合查詢可以是 PG。
start_time	timestamp	開始查詢的時間。
end_time	timestamp	查詢完成的時間。
duration	bigint	查詢花費的時間 (微秒)。
total_partitions	integer	Amazon S3 查詢所需的分割區數量。

欄名稱	資料類型	描述
qualified_partitions	integer	Amazon S3 查詢掃描的分割區數量。
scanned_files	bigint	要掃描的 Amazon S3 檔案數目。
returned_rows	bigint	Amazon S3 查詢的已掃描列數，或是聯合查詢的傳回列數。
returned_bytes	bigint	Amazon S3 查詢的已掃描位元組數，或是聯合查詢的傳回位元組數。
file_format	text	Amazon S3 檔案的檔案格式。
file_location	text	外部資料表的 Amazon S3 位置。
external_query_text	text	聯合查詢的區段層級查詢文字。
warning_message	character(4000)	執行查詢時顯示的警告訊息。
table_name	character(136)	正在操作之步驟的資料表名稱。
is_recursive	character(1)	指示是否對子資料夾遞迴掃描。
is_nested	character(1)	指出是否存取巢狀欄資料類型。
s3list_time	bigint	檔案列出的持續時間，以毫秒為單位。

欄名稱	資料類型	描述
獲取分區時間	long	列出和 Apache Hive 中指定外部物件的分割區所花費 AWS Glue Data Catalog 的時間。

範例查詢

下列查詢會顯示外部查詢詳細資訊。

```
SELECT query_id,
       segment_id,
       start_time,
       end_time,
       total_partitions,
       qualified_partitions,
       scanned_files,
       returned_rows,
       returned_bytes,
       trim(external_query_text) query_text,
       trim(file_location) file_location
FROM sys_external_query_detail
ORDER BY query_id, start_time DESC
LIMIT 2;
```

輸出範例。

```
 query_id | segment_id |         start_time         |          end_time
   | total_partitions | qualified_partitions | scanned_files | returned_rows |
 returned_bytes | query_text | file_location
----------+------------+----------------------------+----------------------------
+------------------+----------------------+--------------+--------------
+----------------+------------+---------------
   763251 |          0 | 2022-02-15 22:32:23.312448 | 2022-02-15 22:32:24.036023 |
          3 |                    3 |             3 |         38203 |        2683414 |
          |
   763254 |          0 | 2022-02-15 22:32:40.17103  | 2022-02-15 22:32:40.839313 |
          3 |                    3 |             3 |         38203 |        2683414 |
          |
```

SYS_EXTERNAL_QUERY_ERROR

您可以查詢系統檢視 SYS_EXTERNAL_QUERY_ERROR，以取得有關 Redshift Spectrum 掃描錯誤的資訊。SYS_EXTERNAL_QUERY_ERROR 會顯示記錄錯誤的範例。預設值是每個查詢 10 個項目。

所有使用者都可看見 SYS_EXTERNAL_QUERY_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生此列之使用者的識別碼。
query_id	bigint	產生此列之查詢的查詢識別碼。
file_location	char(256)	要查詢之資料的位置。
rowid	char(2100)	錯誤在檔案內的位置。`rowid` 部分以 `:` (冒號) 分隔，未來可能會添加其他部分。 `row_offset :row_group :row_id` Row_offset 是檔案中列的偏移量 (以位元組為單位)，針對不支援的檔案格式設定為 `-1`。資料表分為 row_groups，每個群組都有具有不同 row_id 的列。
column_name	char(127)	查詢傳回的欄名稱。
original_value	char(1024)	已查詢原始值。
modified_value	char(1024)	根據查詢中指定的資料處理組態選項傳回的修改值。
觸發條件	char(128)	查詢中指定的資料處理選項。
動作	char(128)	與查詢中指定的資料處理選項相關聯的動作。

欄名稱	資料類型	描述
action_value	char(128)	與查詢中指定的資料處理選項相關聯的動作參數值。
error_code	integer	查詢中指定之資料處理選項的結果代碼。

範例查詢

下列查詢會傳回執行資料處理操作的列清單。

```
SELECT * FROM sys_external_query_error;
```

此查詢會傳回類似以下的結果。

```
  user_id   query_id  file_location                                rowid
column_name           original_value             modified_value       trigger
action               action_value                 error_code
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0
league_name           Barclays Premier League    Barclays Premier Lea UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0
league_nspi           34595                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:1
league_nspi           34151                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2
league_name           Barclays Premier League    Barclays Premier Lea UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2
league_nspi           33223                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3
league_name           Barclays Premier League    Barclays Premier Lea UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3
league_nspi           32808                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
```

```
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:4
league_nspi           32790                      32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:5
league_name           Spanish Primera Division   Spanish Primera Divi UNSPECIFIED
TRUNCATE                                          156
    100     1574007   s3://spectrum-uddh/league/spi_global_rankings.0:6
league_name           Spanish Primera Division   Spanish Primera Divi UNSPECIFIED
TRUNCATE                                          156
```

SYS_INTEGRATION_ACTIVITY

SYS_INTEGRATION_ACTIVITY 會顯示已完成整合執行的相關資訊。

只有超級使用者才能看到 SYS_INTEGRATION_ACTIVITY。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

如需零 ETL 整合的相關資訊，請參閱 Amazon Redshift 管理指南中的使用零 ETL 整合。

資料表欄

欄名稱	資料類型	描述
integration_id	character (128)	與整合關聯的識別碼。
target_database	character (128)	Amazon Redshift 中接收整合資料的資料庫。
source	character (128)	整合的來源資料。可能的類型包括 MySQL 和 PostgreSQL 。
checkpoint_name	character (128)	檢查點複寫 binlog 座標的名稱。
checkpoint_type	character (16)	檢查點的類型。可能的值包括：snapshot、cdc。
checkpoint_bytes	bigint	此檢查點中的位元組數。

欄名稱	資料類型	描述
last_commit_timestamp	timestamp	上次在此檢查點中遞交的時間戳記。
modified_tables	integer	檢查點中修改的資料表數目。
integration_start_time	time	此檢查點的整合開始時間 (UTC)。
integration_end_time	time	此檢查點的整合結束時間 (UTC)。

範例查詢

下列 SQL 命令會顯示整合的記錄。

```
select * from sys_integration_activity;

          integration_id              | target_database | source |
   checkpoint_name                  | checkpoint_type  | checkpoint_bytes |
 last_commit_timestamp   | modified_tables |   integration_start_time   |
 integration_end_time
--------------------------------------+-----------------+--------
+----------------------------------------------+------------------+------------------
+-------------------------+-----------------+---------------------------
+----------------------------
 76b15917-afae-4447-b7fd-08e2a5acce7b |   demo1         | MySQL  | checkpoints/
checkpoint_3_241_3_510.json     |        cdc       |        762       | 2023-05-10
 23:00:14.201 |         1       | 2023-05-10 23:00:45.054265 | 2023-05-10
 23:00:46.339826
 76b15917-afae-4447-b7fd-08e2a5acce7b |   demo1         | MySQL  | checkpoints/
checkpoint_3_16329_3_17839.json |        cdc       |       13488      | 2023-05-11
 01:33:57.411 |         2       | 2023-05-11 02:19:09.440121 | 2023-05-11
 02:19:16.090492
 76b15917-afae-4447-b7fd-08e2a5acce7b |   demo1         | MySQL  | checkpoints/
checkpoint_3_5103_3_5532.json   |        cdc       |        1657      | 2023-05-10
 23:13:14.205 |         2       | 2023-05-10 23:13:23.545487 | 2023-05-10
 23:13:25.652144
```

系統整合 _ 表格狀態變更

系統整合表格狀態變更記錄會顯示有關整合之表格狀態變更記錄檔的詳細資料。

超級用戶可以看到此表中的所有行。

如需詳細資訊，請參閱使用零 ETL 整合。

資料表欄

欄名稱	資料類型	描述
integration_id	character (128)	與整合關聯的識別碼。
database_name	character (128)	Amazon Redshift 資料庫的名稱。
schema_name	character (128)	Amazon Redshift 結構描述的名稱。
table_name	character (128)	資料表的名稱。
新狀態	character (128)	資料表的狀態。可能值為 Synced、ResyncRequired 、ResyncInitiated 、Deleted、Failed 以及 ResyncDeleted 。
table_last_replicated_checkpoint	character (128)	目前同步的記錄座標。
狀態變更原因	character (256)	最後狀態轉換的原因。
record_time	timestamp	更新此記錄的時間 (UTC)。

範例查詢

下列 SQL 命令會顯示整合的記錄。

```
select * from sys_integration_table_state_change;
```

```
            integration_id            | database_name | schema_name | table_name
 | new_state |  table_last_replicated_checkpoint   | state_change_reason |
 record_time
--------------------------------------+---------------+-------------+------------
+-----------+--------------------------------------+---------------------
+----------------------------
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest79   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:39:50.087868
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest56   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:39:45.54005
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest50   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:40:20.362504
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest80t3s | sbtest18   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 19:40:32.544084
 99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb        | sbtest40t3s | sbtest23   |
 Synced    | {"txn_seq":9834,"txn_id":126597515} |                     | 2023-09-20
 15:49:05.186209
```

SYS_LOAD_DETAIL

傳回資訊以追蹤資料載入或對其進行故障診斷。

此檢視會在每一個資料檔案載入至資料庫資料表時記錄其進度。

所有使用者都可看見此檢視。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
query_id	integer	查詢 ID。
file_name	character(256)	要載入的檔案名稱。

欄名稱	資料類型	描述
bytes_scanned	integer	從 Amazon S3 中的檔案掃描的位元組數。
lines_scanned	integer	從載入檔案掃描的行數。此數目可能不符合實際載入的資料列數目。例如，根據 COPY 命令中的 MAXERROR 選項，載入可能掃描但容忍一些錯誤記錄。
record_time	timestamp	上次更新此項目的時間。
splits_scanned	此檔案的分割數目。	此檔案的分割數目。
start_time	timestamp	此檔案處理開始的時間。
end_time	timestamp	此檔案處理完成的時間。

範例查詢

下列範例傳回上次 COPY 操作的詳細資訊。

```
select query_id, trim(file_name) as file, record_time
from sys_load_detail
where query_id = pg_last_copy_id();

 query_id |               file                |          record_time
----------+-----------------------------------+----------------------------
 28554    | s3://dw-tickit/category_pipe.txt | 2013-11-01 17:14:52.648486
(1 row)
```

下列查詢包含 TICKIT 資料庫中全新載入之資料表的項目：

```
select query_id, trim(file_name), record_time
from sys_load_detail
where file_name like '%tickit%' order by query_id;

 query_id |           btrim           |          record_time
----------+---------------------------+----------------------------
 22475    | tickit/allusers_pipe.txt | 2013-02-08 20:58:23.274186
```

```
 22478    | tickit/venue_pipe.txt    | 2013-02-08 20:58:25.070604
 22480    | tickit/category_pipe.txt | 2013-02-08 20:58:27.333472
 22482    | tickit/date2008_pipe.txt | 2013-02-08 20:58:28.608305
 22485    | tickit/allevents_pipe.txt| 2013-02-08 20:58:29.99489
 22487    | tickit/listings_pipe.txt | 2013-02-08 20:58:37.632939
 22593    | tickit/allusers_pipe.txt | 2013-02-08 21:04:08.400491
 22596    | tickit/venue_pipe.txt    | 2013-02-08 21:04:10.056055
 22598    | tickit/category_pipe.txt | 2013-02-08 21:04:11.465049
 22600    | tickit/date2008_pipe.txt | 2013-02-08 21:04:12.461502
 22603    | tickit/allevents_pipe.txt| 2013-02-08 21:04:14.785124
 22605    | tickit/listings_pipe.txt | 2013-02-08 21:04:20.170594

(12 rows)
```

事實上，記錄寫入至此系統檢視的日誌檔案，並不表示已成功遞交載入，做為其包含交易的一部分。若要驗證載入遞交，請查詢 STL_UTILITYTEXT 檢視，並尋找與 COPY 交易相對應的 COMMIT 記錄。例如，此查詢會針對 STL_UTILITYTEXT 根據子查詢聯結 SYS_LOAD_DETAIL 和 STL_QUERY：

```
select l.query_id,rtrim(l.file_name),q.xid
from sys_load_detail l, stl_query q
where l.query_id=q.query
and exists
(select xid from stl_utilitytext where xid=q.xid and rtrim("text")='COMMIT');

 query_id |           rtrim           |  xid
----------+---------------------------+-------
 22600    | tickit/date2008_pipe.txt  | 68311
 22480    | tickit/category_pipe.txt  | 68066
  7508    | allusers_pipe.txt         | 23365
  7552    | category_pipe.txt         | 23415
  7576    | allevents_pipe.txt        | 23429
  7516    | venue_pipe.txt            | 23390
  7604    | listings_pipe.txt         | 23445
 22596    | tickit/venue_pipe.txt     | 68309
 22605    | tickit/listings_pipe.txt  | 68316
 22593    | tickit/allusers_pipe.txt  | 68305
 22485    | tickit/allevents_pipe.txt | 68071
  7561    | allevents_pipe.txt        | 23429
  7541    | category_pipe.txt         | 23415
  7558    | date2008_pipe.txt         | 23428
 22478    | tickit/venue_pipe.txt     | 68065
   526    | date2008_pipe.txt         |  2572
  7466    | allusers_pipe.txt         | 23365
```

```
 22482    | tickit/date2008_pipe.txt  | 68067
 22598    | tickit/category_pipe.txt  | 68310
 22603    | tickit/allevents_pipe.txt | 68315
 22475    | tickit/allusers_pipe.txt  | 68061
   547    | date2008_pipe.txt         |  2572
 22487    | tickit/listings_pipe.txt  | 68072
  7531    | venue_pipe.txt            | 23390
  7583    | listings_pipe.txt         | 23445
(25 rows)
```

SYS_LOAD_ERROR_DETAIL

使用 SYS_LOAD_ERROR_DETAIL 來檢視 COPY 命令錯誤的詳細資料。每一列代表一個 COPY 命令。它包含正在執行和已完成的 COPY 命令。

所有使用者都可看見 SYS_LOAD_ERROR_DETAIL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交副本之使用者的識別碼。
query_id	bigint	副本的查詢識別碼。
transaction_id	bigint	交易識別碼。
session_id	integer	執行副本之程序的處理程序識別碼。
database_name	character(64)	發出複製時，要將使用者連接至其中的資料庫名稱。
table_id	integer	資料表識別碼。
start_time	timestamp	開始複製的時間 (UTC)。
file_name	character(256)	用於載入之輸入檔案的完整路徑。

欄名稱	資料類型	描述
line_number	bigint	載入檔案中發生錯誤的行號。載入 JSON 檔案時，出現錯誤的 JSON 物件的最後一行的行號。
column_name	character(127)	發生錯誤的欄位。
column_type	character(10)	包含錯誤之欄位的資料類型。
column_length	character(10)	欄長度 (如適用)。當資料類型具有限制長度時，會填入此欄位。例如，對於資料類型為 "character(3)" 的欄，此欄將包含值 "3"。
position	integer	欄位中錯誤的位置。
error_code	integer	錯誤代碼。
error_message	character(512)	錯誤的解釋。

範例查詢

下列查詢會顯示特定查詢複製命令的載入錯誤詳細資訊。

```
SELECT query_id,
       table_id,
       start_time,
       trim(file_name) AS file_name,
       trim(column_name) AS column_name,
       trim(column_type) AS column_type,
       trim(error_message) AS error_message
FROM sys_load_error_detail
WHERE query_id = 762949
ORDER BY start_time
LIMIT 10;
```

輸出範例。

```
 query_id | table_id |         start_time         |               file_name
      | column_name | column_type |                 error_message
----------+----------+----------------------------
+-----------------------------------------+------------+------------
+-----------------------------------------------
   762949 |   137885 | 2022-02-15 22:14:46.759151 | s3://load-test/copyfail/
wrong_format_000 | id          | int4        | Invalid digit, Value 'a', Pos 0, Type:
 Integer
   762949 |   137885 | 2022-02-15 22:14:46.759151 | s3://load-test/copyfail/
wrong_format_001 | id          | int4        | Invalid digit, Value 'a', Pos 0, Type:
 Integer
```

SYS_LOAD_HISTORY

使用 SYS_LOAD_HISTORY 來檢視 COPY 命令的詳細資訊。每一列代表一個 COPY 命令，其中包含某些欄位的累計統計資料。它包含正在執行和已完成的 COPY 命令。

所有使用者都可看見 SYS_LOAD_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交副本之使用者的識別碼。
query_id	bigint	副本的查詢識別碼。
transaction_id	bigint	交易識別碼。
session_id	integer	執行副本之程序的處理程序識別碼。
database_name	text	發出操作時，要將使用者連接至其中的資料庫名稱。
status	text	副本的狀態。有效值為 `running`、`completed` 、`aborted`。

欄名稱	資料類型	描述
table_name	text	複製到的資料表的名稱。
start_time	timestamp	複製開始的時間。
end_time	timestamp	複製完成的時間。
duration	bigint	COPY 命令花費的時間 (微秒)。
data_source	text	要複製檔案輸入的 Amazon S3 位置。
file_format	text	來源檔案格式。格式包括 csv、txt、json、avro、orc 或 parquet。
loaded_rows	bigint	複製到資料表的列數。
loaded_bytes	bigint	複製到資料表的位元組數。
source_file_count	integer	來源檔案中的檔案數。
source_file_bytes	bigint	來源檔案中的位元組數。
file_count_scanned	integer	從 Amazon S3 掃描的檔案數。
file_bytes_scanned	bigint	從 Amazon S3 中的檔案掃描的位元組數。
error_count	bigint	錯誤計數。
copy_job_id	bigint	複製任務識別碼。0 表示沒有任務識別碼。

範例查詢

下列查詢顯示特定複製命令的載入列、位元組、資料表和資料來源。

```
SELECT query_id,
       table_name,
       data_source,
       loaded_rows,
       loaded_bytes
FROM sys_load_history
WHERE query_id IN (6389,490791,441663,74374,72297)
ORDER BY query_id,
         data_source DESC;
```

輸出範例。

```
 query_id |    table_name    |                                data_source
              | loaded_rows | loaded_bytes
----------+------------------
+-------------------------------------------------------------------------+-------------
+---------------
     6389 | store_returns    | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
store_returns/    |   287999764 | 1196240296158
    72297 | web_site         | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
web_site/         |          54 |         43808
    74374 | ship_mode        | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
ship_mode/        |          20 |          1320
   441663 | income_band      | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
income_band/      |          20 |          2152
   490791 | customer_address | s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/
customer_address/ |     6000000 |     722924305
```

下列查詢顯示複製命令的載入列、位元組、資料表和資料來源。

```
SELECT query_id,
       table_name,
       data_source,
       loaded_rows,
       loaded_bytes
FROM sys_load_history
ORDER BY query_id DESC
LIMIT 10;
```

輸出範例。

```
 query_id |       table_name       |                                  data_source
                          | loaded_rows |  loaded_bytes
----------+------------------------
+-------------------------------------------------------------------------------
+-------------+-----------------
   491058 | web_site               | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_site/               |          54 |           43808
   490947 | web_sales              | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_sales/              |   720000376 |  22971988122819
   490923 | web_returns            | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_returns/            |    71997522 |     96597496325
   490918 | web_page               | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/web_page/               |        3000 |            1320
   490907 | warehouse              | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/warehouse/              |          20 |            1320
   490902 | time_dim               | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/time_dim/               |       86400 |            1320
   490876 | store_sales            | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/store_sales/            |  2879987999 | 151666241887933
   490870 | store_returns          | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/store_returns/          |   287999764 |   1196405607941
   490865 | store                  | s3://load-test/data-sources/tpcds/2.8.0/
textfile/1T/store/                  |        1002 |          365507
```

下列查詢顯示複製命令的每日載入列和位元組。

```
SELECT date_trunc('day',start_time) AS exec_day,
       SUM(loaded_rows) AS loaded_rows,
       SUM(loaded_bytes) AS loaded_bytes
FROM sys_load_history
GROUP BY exec_day
ORDER BY exec_day DESC;
```

輸出範例。

```
      exec_day       | loaded_rows |   loaded_bytes
---------------------+-------------+------------------
 2022-01-20 00:00:00 |  6347386005 |  258329473070606
 2022-01-19 00:00:00 | 19042158015 |  775198502204572
 2022-01-18 00:00:00 | 38084316030 | 1550294469446883
 2022-01-17 00:00:00 | 25389544020 | 1033271084791724
 2022-01-16 00:00:00 | 19042158015 |  775222736252792
```

```
2022-01-15 00:00:00 | 19834245387 |  798122849155598
2022-01-14 00:00:00 | 75376544688 | 3077040926571384
```

SYS_MV_REFRESH_HISTORY

結果包括所有具體化視觀表之重新整理歷史記錄的相關資訊。結果包括重新整理類型 (例如手動或自動)，以及最近重新整理的狀態。

所有使用者都可看見 SYS_MV_REFRESH_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交重新整理的使用者識別碼。
session_id	integer	執行具體化視觀表之程序的處理程序識別碼。
transaction_id	bigint	交易識別碼。
database_name	char(128)	包含具體化視觀表的資料庫。
schema_name	char(128)	具體化視觀表的結構描述。
mv_id	bigint	具體化視觀表的物件 ID。
mv_name	char(128)	具體化視觀表名稱。
refresh_type	char(32)	重新整理的類型，例如手動或自動。
status	text	重新整理的狀態。如需狀態的詳細資訊，請參閱 SVL_MV_REFRESH_STATUS 的狀態欄。
start_time	timestamp	重新整理的開始時間。

欄名稱	資料類型	描述
end_time	timestamp	重新整理的結束時間。
duration	bigint	重新整理具體化視觀表所花費的時間 (以微秒為單位)。

範例查詢

下列查詢顯示具體化視觀表的重新整理歷史記錄。

```
SELECT user_id,
     session_id,
     transaction_id,
     database_name,
     schema_name,
     mv_id,
     mv_name,
     refresh_type,
     status,
     start_time,
     end_time,
     duration
     from sys_mv_refresh_history;
```

此查詢會傳回下列範例輸出：

```
 user_id | session_id | transaction_id | database_name | schema_name                |
 mv_id  |  mv_name           |  refresh_type  |  status
         |  start_time                |  end_time                  |  duration
---------+------------+----------------+---------------
+----------------------------+--------+--------------------+----------------
+-------------------------------------------------+----------------------------
+----------------------------+-----------
       1 | 1073815659 |          15066 | dev           | test_stl_mv_refresh_schema |
 203762 | mv_incremental     | Manual         | MV was already updated
         | 2023-10-26 15:59:20.952179 | 2023-10-26 15:59:20.952866 |      687
       1 | 1073815659 |          15068 | dev           | test_stl_mv_refresh_schema |
 203771 | mv_nonincremental  | Manual         | MV was already updated
         | 2023-10-26 15:59:21.008049 | 2023-10-26 15:59:21.008658 |      609
```

```
      1 | 1073815659 |          15070 | dev           | test_stl_mv_refresh_schema |
 203779 | mv_refresh_error   | Manual         | MV was already updated
        | 2023-10-26 15:59:21.064252 | 2023-10-26 15:59:21.064885 |      633
      1 | 1073815659 |          15074 | dev           | test_stl_mv_refresh_schema
 | 203762 | mv_incremental     | Manual         | Refresh successfully updated MV
 incrementally   | 2023-10-26 15:59:29.693329 | 2023-10-26 15:59:43.482842 | 13789513
      1 | 1073815659 |          15076 | dev           | test_stl_mv_refresh_schema |
 203771 | mv_nonincremental  | Manual         | Refresh successfully recomputed MV from
 scratch | 2023-10-26 15:59:43.550184 | 2023-10-26 15:59:47.880833 |  4330649
      1 | 1073815659 |          15078 | dev           | test_stl_mv_refresh_schema |
 203779 | mv_refresh_error   | Manual         | Refresh failed due to an internal error
        | 2023-10-26 15:59:47.949052 | 2023-10-26 15:59:52.494681 |  4545629
(6 rows)
```

SYS_MV_STATE

結果包括所有具體化視觀表狀態的相關資訊。它包括基底資料表資訊、結構描述屬性，以及有關最近事件的資訊，例如卸除欄。

所有使用者都可看見 SYS_MV_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	bigint	建立事件的使用者 ID。
transaction_id	bigint	事件的交易 ID。
database_name	char(128)	包含具體化視觀表的資料庫。
event_desc	char(500)	提示狀態變更的事件。範例值包括下列各項： • 已變更資料欄類型 • 已捨棄資料欄 • 已重新命名資料欄 • 已變更結構描述名稱 • 小型資料表轉換

欄名稱	資料類型	描述
		• TRUNCATE • Vacuum 請注意，此欄還有其他可能的值。
start_time	timestamp	事件的開始時間。
base_table_database_name	char(128)	基底資料表的資料庫名稱。
base_table_schema	char(128)	基底資料表的結構描述。
base_table_name	char(128)	基底資料表的名稱。
mv_schema	char(128)	具體化視觀表的結構描述。
mv_name	char(128)	具體化視觀表的名稱。
state	character(32)	具體化視觀表的變更後狀態如下： • 重新運算 • 無法重新整理

範例查詢

下列查詢顯示具體化視觀表的狀態。

```
select * from sys_mv_state;
```

此查詢會傳回下列範例輸出：

```
user_id | transaction_id | database_name | event_desc                  | start_time
             | base_table_database_name | base_table_schema | base_table_name     |
mv_schema  | mv_name                    | state
```

```
---------+----------------+---------------+----------------------------
+----------------------------+-------------------------+------------------
+---------------------+-------------+----------------------------+--------------
 106     | 12720          | tickit_db     | TRUNCATE                    | 2023-07-26
 14:59:12.788268 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_a1f3f862 | Recompute
 106     | 12724          | tickit_db     | ALTER TABLE ALTER DISTSTYLE | 2023-07-26
 14:59:51.409014 | tickit_db                | mv_schema         | test_table_58102435 |
 mv_schema   | materialized_view_ca746631 | Recompute
 106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
 14:59:12.822928 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5750a8d4 | Unrefreshable
 106     | 12727          | tickit_db     | Table was renamed           | 2023-07-26
 15:00:08.051244 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5750a8d4 | Unrefreshable
 106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
 14:59:12.857755 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5750a8d4 | Unrefreshable
 106     | 12727          | tickit_db     | Table was renamed           | 2023-07-26
 15:00:08.051358 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5ef0d754 | Unrefreshable
 106     | 12720          | tickit_db     | TRUNCATE                    | 2023-07-26
 14:59:12.788159 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5750a8d4 | Recompute
 106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
 14:59:12.857799 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_a1f3f862 | Unrefreshable
 106     | 12720          | tickit_db     | TRUNCATE                    | 2023-07-26
 14:59:12.788327 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_5ef0d754 | Recompute
 106     | 12727          | tickit_db     | ALTER TABLE ALTER SORTKEY   | 2023-07-26
 15:00:08.006235 | tickit_db                | mv_schema         | test_table_58102435 |
 mv_schema   | materialized_view_ca746631 | Recompute
 106     | 12720          | tickit_db     | Column was renamed          | 2023-07-26
 14:59:12.82297  | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_a1f3f862 | Unrefreshable
 106     | 12727          | tickit_db     | Table was renamed           | 2023-07-26
 15:00:08.051321 | tickit_db                | mv_schema         | test_table_95d6d861 |
 mv_schema   | materialized_view_a1f3f862 | Unrefreshable
```

SYS_PROCEDURE_CALL

使用 SYS_PROCEDURE_CALL 檢視來取得預存程序呼叫的相關資訊，包括開始時間、結束時間、預存程序呼叫的狀態，以及巢狀預存程序呼叫的呼叫階層。每次預存程序呼叫會接收查詢 ID。

所有使用者都可看見 SYS_PROCEDURE_CALL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
session_user_id	integer	使用者的識別碼，該使用者建立工作階段，並且是最上層預存程序呼叫的叫用者。
security_user_id	integer	使用者的識別碼，該使用者的權限用於執行預存程序中的陳述式。如果預存程序是 DEFINER，那麼這將是預存程序的擁有者 user_id。
query_id	integer	預存程序呼叫的查詢識別碼。
query_text	char(4000)	預存程序呼叫查詢的文字。
start_time	timestamp	查詢開始執行的時間 (以 UTC 為單位)。例如，時間戳記使用六位數精確度來表示小數秒。2009-06-12 11:29:19.131358.
end_time	timestamp	查詢完成執行的時間 (以 UTC 為單位)。時間戳記會使用六位數精確度來表示小數秒，例如：2009-06-12 11:29:19.131358。
status	char(10)	預存程序呼叫的狀態。系統停止或使用者取消預存程序時，

欄名稱	資料類型	描述
		會取消該值。如果預存程序呼叫執行到完成，則值為成功。
caller_procedure_query_id	integer	如果預存程序呼叫由另一個預存程序呼叫所調用，則此欄包含外層呼叫的查詢 ID。否則此欄位為 NULL。

範例查詢

下列查詢會傳回巢狀預存程序呼叫階層。

```
select query_id, datediff(seconds, start_time, end_time) as elapsed_time, status,
 trim(query_text) as call, caller_procedure_query_id from sys_procedure_call;
```

輸出範例。

```
 query_id | elapsed_time | status  |                        call                        |
 caller_procedure_query_id
----------+--------------+---------+----------------------------------------------------
+---------------------------
     3087 |           18 | success | CALL proc_bd906c98c45443ffa165e9552056902d(1)    |
          3085
     3085 |           18 | success | CALL proc_bd906c98c45443ffa165e9552056902d_2(1); |

(2 rows)
```

SYS_PROCEDURE_MESSAGES

所有使用者都可看見 SYS_PROCEDURE_MESSAGES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
transaction_id	bigint	交易識別碼。

欄名稱	資料類型	描述
query_id	integer	預存程序呼叫的查詢識別碼。
record_time	timestamp	產生訊息的時間 (以 UTC 為單位)。
log_level	char(10)	產生訊息的記錄層級。可能的值為 LOG、INFO、NOTICE、WARNING 和 EXCEPTION。
message	char(1024)	產生訊息的文字。
line_number	integer	產生訊息的行號。

範例查詢

下列查詢顯示 SYS_PROCEDURE_MESSAGES 的範例輸出。

```
select transaction_id, query_id, record_time, log_level, trim(message), line_number
 from sys_procedure_messages;
```

```
transaction_id | query_id |        record_time         | log_level |           btrim
        | line_number
---------------+----------+----------------------------+-----------
+--------------------------+-------------
     25267     |   80562  | 2023-07-17 14:38:31.910136 |   NOTICE  |
 test_notice_msg_b9f1e749  |     8
     25267     |   80562  | 2023-07-17 14:38:31.910002 |    LOG    |
 test_log_msg_833c7420     |     6
     25267     |   80562  | 2023-07-17 14:38:31.910111 |    INFO   |
 test_info_msg_651373d9    |     7
     25267     |   80562  | 2023-07-17 14:38:31.910154 |   WARNING |
 test_warning_msg_831c5747 |     9
(4 rows)
```

SYS_QUERY_DETAIL

使用 SYS_QUERY_DETAIL 來檢視步驟層級的查詢詳細資訊。每一列代表特定 WLM 查詢的一個步驟以及詳細資訊。此檢視包含許多類型的查詢，例如 DDL、DML 和公用程式命令 (例如，複製和卸載)。根據查詢類型的不同，某些欄可能不相關。例如，external_scanned_bytes 與內部資料表不相關。

所有使用者都可看見 SYS_QUERY_DETAIL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	查詢識別碼。
child_query_sequence	integer	重寫使用者查詢的順序，從 1 開始。
stream_id	integer	查詢串流的資料流識別碼。
segment_id	integer	查詢執行區段的區段識別碼。
step_id	integer	區段中的步驟識別碼。
step_name	text	區段中的步驟名稱。可能的值為aggregate broadcast 、delete、distribut e 、hash、hashjoin、insert、li 、unique、和window。
table_id	integer	永久資料表掃描的資料表識別碼。
table_name	character(136)	正在操作之步驟的資料表名稱。

欄名稱	資料類型	描述
is_rrscan	character	指示步驟是否為掃描步驟的值。True (t) 表示使用了範圍限制掃描。
start_time	timestamp	查詢步驟開始的時間。
end_time	timestamp	查詢步驟完成的時間。
duration	bigint	步驟花費的時間 (微秒)。
提醒	text	提醒事件的說明。
input_bytes	bigint	目前步驟的輸入位元組。
input_rows	bigint	目前步驟的輸入列。
output_bytes	bigint	目前步驟的輸出位元組。
output_rows	bigint	目前步驟的輸出列。
blocks_read	bigint	步驟讀取的區塊數目。
blocks_write	bigint	步驟寫入的區塊數目。
local_read_IO	bigint	從本機磁碟快取讀取的區塊數目。
remote_read_IO	bigint	從遠端讀取的區塊數。
source	text	已掃描的資料庫物件類型。只有當列的 step_name 值為 scan 時，此欄才有值。
data_skewness	integer	輸出列在所有步驟之間分佈的偏態。它是一個在 0% 到 100% 的範圍內的數字。數字越大，分佈越不平衡。

欄名稱	資料類型	描述
time_skewness	integer	所有步驟之間執行時間分佈的偏態。它是一個在 0% 到 100% 的範圍內的數字。數字越大，分佈越不平衡。
is_active	character	步驟層級查詢的狀態。可能的值是 't'，顯示步驟正在執行中；或 'f'，表示步驟已完成執行。
spilled_block_local_disk	bigint	溢出到本機磁碟的區塊數目。
spilled_block_remote_disk	bigint	溢出到 Amazon Simple Storage Service 的區塊數。
step_attribute	character(64)	包含相關聯步驟的資訊。掃描步驟的可能值：`multi-dimensional` 。

範例查詢

下列範例會傳回 SYS_QUERY_DETAIL 的輸出。

以下查詢顯示步驟層級的查詢中繼資料詳細資訊，包括步驟名稱、input_bytes、output_bytes、input_rows、output_rows。

```
SELECT query_id,
       child_query_sequence,
       stream_id,
       segment_id,
       step_id,
       trim(step_name) AS step_name,
       duration,
       input_bytes,
       output_bytes,
       input_rows,
       output_rows
FROM sys_query_detail
```

```
WHERE query_id IN (193929)
ORDER BY query_id,
         stream_id,
         segment_id,
         step_id DESC;
```

輸出範例。

```
 query_id | child_query_sequence | stream_id | segment_id | step_id | step_name  |
 duration     | input_bytes | output_bytes | input_rows | output_rows
----------+----------------------+-----------+------------+---------+------------
+-----------------+-------------+--------------+------------+-------------
   193929 |                    2 |         0 |          0 |       3 | hash       |
      37144 |           0 |      9350272 |          0 |      292196
   193929 |                    5 |         0 |          0 |       3 | hash       |
       9492 |           0 |        23360 |          0 |        1460
   193929 |                    1 |         0 |          0 |       3 | hash       |
      46809 |           0 |      9350272 |          0 |      292196
   193929 |                    4 |         0 |          0 |       2 | return     |
       7685 |           0 |          896 |          0 |         112
   193929 |                    1 |         0 |          0 |       2 | project    |
      46809 |           0 |            0 |          0 |      292196
   193929 |                    2 |         0 |          0 |       2 | project    |
      37144 |           0 |            0 |          0 |      292196
   193929 |                    5 |         0 |          0 |       2 | project    |
       9492 |           0 |            0 |          0 |        1460
   193929 |                    3 |         0 |          0 |       2 | return     |
      11033 |           0 |        14336 |          0 |         112
   193929 |                    2 |         0 |          0 |       1 | project    |
      37144 |           0 |            0 |          0 |      292196
   193929 |                    1 |         0 |          0 |       1 | project    |
      46809 |           0 |            0 |          0 |      292196
   193929 |                    5 |         0 |          0 |       1 | project    |
       9492 |           0 |            0 |          0 |        1460
   193929 |                    3 |         0 |          0 |       1 | aggregate  |
      11033 |           0 |       201488 |          0 |          14
   193929 |                    4 |         0 |          0 |       1 | aggregate  |
       7685 |           0 |        28784 |          0 |          14
   193929 |                    5 |         0 |          0 |       0 | scan       |
       9492 |           0 |        23360 |     292196 |        1460
   193929 |                    4 |         0 |          0 |       0 | scan       |
       7685 |           0 |         1344 |        112 |         112
```

```
   193929 |                   2 |         0 |          0 |       0 | scan       |
      37144 |           0 |      7304900 |     292196 |      292196
   193929 |                   3 |         0 |          0 |       0 | scan       |
      11033 |           0 |        13440 |        112 |         112
   193929 |                   1 |         0 |          0 |       0 | scan       |
      46809 |           0 |      7304900 |     292196 |      292196
   193929 |                   5 |         0 |          0 |      -1 |            |
       9492 |       12288 |           0 |          0 |          0
   193929 |                   1 |         0 |          0 |      -1 |            |
      46809 |       16384 |           0 |          0 |          0
   193929 |                   2 |         0 |          0 |      -1 |            |
      37144 |       16384 |           0 |          0 |          0
   193929 |                   4 |         0 |          0 |      -1 |            |
       7685 |       28672 |           0 |          0 |          0
   193929 |                   3 |         0 |          0 |      -1 |            |
      11033 |      114688 |           0 |          0 |          0
```

若要檢視資料庫中的資料表，從最常用到最少使用的順序，請使用下列範例。用您自己的資料庫取代 *sample_data_dev*。請注意，此查詢將在建立叢集時開始計算查詢，但是當資料倉儲缺少空間時，系統檢視資料不會儲存。

```
SELECT table_name, COUNT (DISTINCT query_id)
FROM SYS_QUERY_DETAIL
WHERE table_name LIKE 'sample_data_dev%'
GROUP BY table_name
ORDER BY COUNT(*) DESC;

+--------------------------------+-------+
|           table_name           | count |
+--------------------------------+-------+
| sample_data_dev.tickit.venue   |     4 |
| sample_data_dev.myunload1.venue |     3 |
| sample_data_dev.tickit.listing  |     1 |
| sample_data_dev.tickit.category |     1 |
| sample_data_dev.tickit.users    |     1 |
| sample_data_dev.tickit.date     |     1 |
| sample_data_dev.tickit.sales    |     1 |
| sample_data_dev.tickit.event    |     1 |
+--------------------------------+-------+
```

SYS_QUERY_HISTORY

使用 SYS_QUERY_HISTORY 來檢視使用者查詢的詳細資料。每一列代表一個使用者查詢，其中包含某些欄位的累計統計資料。此檢視包含許多類型的查詢，例如資料定義語言 (DDL)、資料處理語言 (DML)、複製、卸載和 Amazon Redshift Spectrum。它包含正在執行和已完成的查詢。

所有使用者都可看見 SYS_QUERY_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	查詢識別碼。
query_label	character(320)	查詢的簡短名稱。
transaction_id	bigint	交易識別碼。
session_id	integer	執行查詢之程序的處理程序識別碼。
database_name	character(128)	當查詢發出時，要將使用者連接至其中的資料庫名稱。
query_type	character(32)	查詢類型，例如「選取」、「插入」、「更新」、「卸載」、「複製」、「命令」、「DDL」、「公用程式」、「CTA」和「其他」。
status	character(10)	查詢的狀態。有效值：planning、queued、running、returning、failed、canceled 和 success。
result_cache_hit	Boolean	指出查詢是否符合結果快取。

欄名稱	資料類型	描述
start_time	timestamp	開始查詢的時間。
end_time	timestamp	查詢完成的時間。
elapsed_time	bigint	查詢所花費的總時間 (微秒)。
queue_time	bigint	服務類別查詢佇列所花費的總時間 (微秒)。
execution_time	bigint	服務類別中執行的總時間 (微秒)。
error_message	character(512)	查詢失敗的原因。
returned_rows	bigint	傳回給用戶端的列數。
returned_bytes	bigint	傳回給用戶端的位元組數。
query_text	character(4000)	查詢字串。此字串可能會被截斷。
redshift_version	character(256)	查詢執行時的 Amazon Redshift 版本。
usage_limit	character(150)	查詢達到的使用限制 ID 清單。
compute_type	varchar(32)	指出查詢是執行於主要叢集或並行擴展叢集。可能值為 `primary` (在主要叢集上執行查詢)、`secondary` (在次要叢集上執行查詢)，或 `primary-scale` (在並行叢集上執行查詢)。僅適用已佈建叢集。
compile_time	bigint	編譯查詢所花費的總時間 (微秒)。

欄名稱	資料類型	描述
planning_time	bigint	規劃查詢所花費的總時間 (微秒)。
lock_wait_time	bigint	等待關係鎖定所花費的總時間 (微秒)。

範例查詢

下列查詢會傳回執行中和佇列中的查詢。

```
SELECT user_id,
       query_id,
       transaction_id,
       session_id,
       status,
       trim(database_name) AS database_name,
       start_time,
       end_time,
       result_cache_hit,
       elapsed_time,
       queue_time,
       execution_time
FROM sys_query_history
WHERE status IN ('running','queued')
ORDER BY start_time;
```

輸出範例。

```
 user_id | query_id | transaction_id | session_id | status  | database_name |
 start_time         |          end_time          | result_cache_hit | elapsed_time |
 queue_time | execution_time
---------+----------+----------------+------------+---------+---------------
+---------------------------+---------------------------+-----------------
+--------------+------------+----------------
     101 |   760705 |         852337 | 1073832321 | running | tpcds_1t      |
 2022-02-15 19:03:19.67849 | 2022-02-15 19:03:19.739811 | f               |
 61321 |          0 |              0
```

下列查詢會傳回特定查詢的查詢開始時間、結束時間、佇列時間、經歷時間、計劃時間及其他中繼資料。

```
SELECT user_id,
       query_id,
       transaction_id,
       session_id,
       status,
       trim(database_name) AS database_name,
       start_time,
       end_time,
       result_cache_hit,
       elapsed_time,
       queue_time,
       execution_time,
       planning_time,
       trim(query_text) as query_text
FROM sys_query_history
WHERE query_id = 3093;
```

輸出範例。

```
user_id | query_id | transaction_id | session_id |   status   | database_name |
   start_time         |          end_time          | result_cache_hit | elapsed_time |
 queue_time | execution_time | planning_time | query_text
--------+----------+----------------+------------+------------
+---------------+----------------------------+----------------------------
+------------------+--------------+------------+----------------+---------------
+-------------------------------------
    106 |     3093 |          11759 | 1073750146 | success    | dev           |
 2023-03-16 16:53:17.840214 | 2023-03-16 16:53:18.106588 | f                |
 266374 |          0 |         105725 |        136589 | select count(*) from item;
```

下列查詢會列出10 個最近的 SELECT 查詢。

```
SELECT query_id,
       transaction_id,
       session_id,
       start_time,
       elapsed_time,
       queue_time,
       execution_time,
```

```
       returned_rows,
       returned_bytes
FROM sys_query_history
WHERE query_type = 'SELECT'
ORDER BY start_time DESC limit 10;
```

輸出範例。

```
 query_id | transaction_id | session_id |         start_time         | elapsed_time |
 queue_time | execution_time | returned_rows | returned_bytes
----------+----------------+------------+----------------------------+--------------
+------------+----------------+---------------+----------------
   526532 |          61093 | 1073840313 | 2022-02-09 04:43:24.149603 |       520571 |
        0 |         481293 |              1 |           3794
   526520 |          60850 | 1073840313 | 2022-02-09 04:38:27.24875  |       635957 |
        0 |         596601 |              1 |           3679
   526508 |          60803 | 1073840313 | 2022-02-09 04:37:51.118835 |       563882 |
        0 |         503135 |              5 |          17216
   526505 |          60763 | 1073840313 | 2022-02-09 04:36:48.636224 |       649337 |
        0 |         589823 |              1 |            652
   526478 |          60730 | 1073840313 | 2022-02-09 04:36:11.741471 |     14611321 |
        0 |       14544058 |              0 |              0
   526467 |          60636 | 1073840313 | 2022-02-09 04:34:11.91463  |     16711367 |
        0 |       16633767 |              1 |            575
   511617 |         617946 | 1074009948 | 2022-01-20 06:21:54.44481  |      9937090 |
        0 |        9899271 |            100 |          12500
   511603 |         617941 | 1074259415 | 2022-01-20 06:21:45.71744  |      8065081 |
        0 |        7582500 |            100 |           8889
   511595 |         617935 | 1074128320 | 2022-01-20 06:21:44.030876 |      1051270 |
        0 |        1014879 |              1 |             72
   511584 |         617931 | 1074030019 | 2022-01-20 06:21:42.764088 |       609033 |
        0 |         485887 |            100 |           8438
```

下列查詢會顯示每日選擇查詢計數和平均查詢經歷時間。

```
SELECT date_trunc('day',start_time) AS exec_day,
       status,
       COUNT(*) AS query_cnt,
       AVG(datediff (microsecond,start_time,end_time)) AS elapsed_avg
FROM sys_query_history
WHERE query_type = 'SELECT'
AND start_time >= '2022-01-14'
```

```
AND start_time <= '2022-01-18'
GROUP BY exec_day,
         status
ORDER BY exec_day,
         status;
```

輸出範例。

```
      exec_day       | status  | query_cnt | elapsed_avg
---------------------+---------+-----------+------------
 2022-01-14 00:00:00 | success |      5253 |  56608048
 2022-01-15 00:00:00 | success |      7004 |  56995017
 2022-01-16 00:00:00 | success |      5253 |  57016363
 2022-01-17 00:00:00 | success |      5309 |  55236784
 2022-01-18 00:00:00 | success |      8092 |  54355124
```

下列查詢會顯示日常查詢的經歷時間效能。

```
SELECT distinct date_trunc('day',start_time) AS exec_day,
       query_count.cnt AS query_count,
       Percentile_cont(0.5) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P50_runtime,
       Percentile_cont(0.8) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P80_runtime,
       Percentile_cont(0.9) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P90_runtime,
       Percentile_cont(0.99) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS P99_runtime,
       Percentile_cont(1.0) within group(ORDER BY elapsed_time) OVER (PARTITION BY
 exec_day) AS max_runtime
FROM sys_query_history
LEFT JOIN (SELECT  date_trunc('day',start_time) AS day, count(*) cnt
           FROM sys_query_history
           WHERE query_type = 'SELECT'
           GROUP by 1) query_count
ON date_trunc('day',start_time) = query_count.day
WHERE query_type = 'SELECT'
ORDER BY exec_day;
```

輸出範例。

```
      exec_day       | query_count | p50_runtime | p80_runtime | p90_runtime |
 p99_runtime  | max_runtime
--------------------+-------------+-------------+-------------+-------------
+-------------+-------------
 2022-01-14 00:00:00 |        5253 |  16816922.0 |  69525096.0 | 158524917.8 |
 486322477.52 | 1582078873.0
 2022-01-15 00:00:00 |        7004 |  15896130.5 |  71058707.0 | 164314568.9 |
 500331542.07 | 1696344792.0
 2022-01-16 00:00:00 |        5253 |  15750451.0 |  72037082.2 | 159513733.4 |
 480372059.24 | 1594793766.0
 2022-01-17 00:00:00 |        5309 |  15394513.0 |  68881393.2 | 160254700.0 |
 493372245.84 | 1521758640.0
 2022-01-18 00:00:00 |        8092 |  15575286.5 |  68485955.4 | 154559572.5 |
 463552685.39 | 1542783444.0
 2022-01-19 00:00:00 |        5860 |  16648747.0 |  72470482.6 | 166485138.2 |
 492038228.67 | 1693483241.0
 2022-01-20 00:00:00 |        1751 |  15422072.0 |  69686381.0 | 162315385.0 |
 497066615.00 | 1439319739.0
 2022-02-09 00:00:00 |          13 |   6382812.0 |  17616161.6 |  21197988.4 |
 23021343.84 |   23168439.0
```

下列查詢顯示查詢類型分佈。

```
SELECT query_type,
       COUNT(*) AS query_count
FROM sys_query_history
GROUP BY query_type
ORDER BY query_count DESC;
```

輸出範例。

```
 query_type | query_count
------------+-------------
 UTILITY    |      134486
 SELECT     |       38537
 DDL        |        4832
 OTHER      |         768
 LOAD       |         768
 CTAS       |         748
 COMMAND    |          92
```

SYS_QUERY_TEXT

使用 SYS_QUERY_TEXT 來檢視所有查詢的查詢文字。每一列代表查詢的查詢文字，最多 4000 個字元，以序列號 0 開始。當查詢陳述式包含超過 4000 個字元時，會藉由遞增每個列的序號來記錄陳述式的其他列。此檢視會記錄所有使用者查詢文字，例如 DDL、公用程式、Amazon Redshift 查詢，以及僅限潛在客戶節點的查詢。

所有使用者都可看見 SYS_QUERY_TEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	查詢識別碼。
transaction_id	bigint	與陳述式關聯的交易 ID。
session_id	integer	執行查詢之工作階段的處理程序識別碼。
start_time	timestamp	查詢開始的時間。
sequence	integer	當單一陳述式包含不只 4000 個字元時，會將該陳述式的其他列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character (4000)	以 4000 個字元為單位遞增的 SQL 查詢文字。此欄位可能包含特殊字元，例如反斜線 () 和換行符號 ()。

範例查詢

下列查詢會傳回執行中和佇列中的查詢。

```
SELECT user_id,
 query_id,
 transaction_id,
 session_id, start_time,
 sequence, trim(text) as text from sys_query_text
 ORDER BY sequence;
```

輸出範例。

```
 user_id | query_id | transaction_id | session_id |        start_time          |
 sequence |                                                        text
--------+----------+----------------+------------+----------------------------
+----------
+-----------------------------------------------------------------------------------------------------------
   100  |     4    |       1396     | 1073750220 | 2023-04-28 16:44:55.887184 |
   0    | SELECT trim(text) as text, sequence FROM sys_query_text WHERE query_id =
 pg_last_query_id() AND user_id > 1 AND start
_time > '2023-04-28 16:44:55.922705+00:00'::timestamp order by sequence;
```

下列查詢會傳回已從資料庫中的群組授與或撤銷的許可。

```
SELECT
    SPLIT_PART(text, ' ', 1) as grantrevoke,
    SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'GROUP'))), ' ', 2) as group,
    SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), ' '))), 'ON', 1) as type,
    SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'ON'))), ' ', 2) || ' ' ||
 SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'ON'))), ' ', 3) as entity
FROM SYS_QUERY_TEXT
WHERE (text LIKE 'GRANT%' OR text LIKE 'REVOKE%') AND text LIKE '%GROUP%';

+-------------+----------+--------+----------+
| grantrevoke |  group   |  type  |  entity  |
+-------------+----------+--------+----------+
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | USAGE  | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
+-------------+----------+--------+----------+
```

SYS_RESTORE_LOG

使用 SYS_RESTORE_LOG 在傳統調整為 RA3 節點大小期間，監控叢集中每個資料表的遷移進度。它會在調整大小作業期間擷取資料遷移的歷史輸送量。如需有關傳統調整為 RA3 節點大小的詳細資訊，請參閱傳統調整大小。

只有超級使用者才能看到 SYS_RESTORE_LOG。

資料表欄

欄名稱	資料類型	描述
event_time	timestamp	指出日誌項目何時記錄的時間戳記。
database_name	char(128)	資料庫的名稱。
schema_name	char(128)	結構描述的名稱。
table_name	char(128)	資料表的名稱。
table_id	integer	資料表的 ID。
動作	char(128)	輸入時所採取的動作。值可包括：已啟動移轉、檢查點、已繼續、已完成、已取消或重設。
table_size	long	資料表的大小
total_data_processed	long	到目前為止，資料表處理的資料大小 (以 MB 為單位)。
delta_data_processed	long	自上次 event_time 更新後所處理的資料大小，以 MB 為單位。這可協助您判斷自先前記錄的時間間隔後，已處理多少資料。您可以將其與 table_size 進行比較，以了解資料處理的速度。

欄名稱	資料類型	描述
message	char(512)	動作欄中值的詳細說明。
redistribution_type	char(32)	資料表的重新發佈類型。KEY 轉換或 EVEN 重新平衡任務。如需分佈樣式的詳細資訊，請參閱分佈樣式。

範例查詢

下列查詢會使用 SYS_RESTORE_LOG 計算資料處理的輸送量。

```
SELECT
    ROUND(sum(delta_data_processed) / 1024.0, 2) as data_processed_gb,
    ROUND(datediff(sec, min(event_time), max(event_time)) / 3600.0, 2) as duration_hr,
    ROUND(data_processed_gb/duration_hr, 2) as throughput_gb_per_hr
from sys_restore_log;
```

輸出範例。

```
 data_processed_gb | duration_hr | throughput_gb_per_hr
-------------------+-------------+----------------------
              0.91 |        8.37 |                 0.11
(1 row)
```

顯示所有重新分配類型的下列查詢。

```
SELECT * from sys_restore_log ORDER BY event_time;
```

```
 database_name |     schema_name      |      table_name      | table_id |
 action             | total_data_processed | delta_data_processed |         event_time
        | table_size | message |   redistribution_type
---------------+----------------------+----------------------+----------
+-----------------------------+----------------------+----------------------
+-----------------------------+------------+---------+--------------------------
 dev            | schemaaaa877096d844d | customer_key         |   106424 |
 Redistribution started      |                    0 |                      | 2024-01-05
 02:18:00.744977 |        325 |         | Restore Distkey Table
```

```
 dev           | schemaaaa877096d844d | dp30907_t2_autokey   |   106430 |
 Redistribution started      |                    0 |                       | 2024-01-05
 02:18:02.756675 |         90 |         | Restore Distkey Table
 dev           | schemaaaa877096d844d | dp30907_t2_autokey   |   106430 |
 Redistribution completed    |                   90 |                    90 | 2024-01-05
 02:23:30.643718 |         90 |         | Restore Distkey Table
 dev           | schemaaaa877096d844d | customer_key         |   106424 |
 Redistribution completed    |                  325 |                   325 | 2024-01-05
 02:23:45.998249 |        325 |         | Restore Distkey Table
 dev           | schemaaaa877096d844d | dp30907_t1_even      |   106428 |
 Redistribution started      |                    0 |                       | 2024-01-05
 02:23:46.083849 |         30 |         | Rebalance Disteven Table
 dev           | schemaaaa877096d844d | dp30907_t5_auto_even |   106436 |
 Redistribution started      |                    0 |                       | 2024-01-05
 02:23:46.855728 |         45 |         | Rebalance Disteven Table
 dev           | schemaaaa877096d844d | dp30907_t5_auto_even |   106436 |
 Redistribution completed    |                   45 |                    45 | 2024-01-05
 02:24:16.343029 |         45 |         | Rebalance Disteven Table
 dev           | schemaaaa877096d844d | dp30907_t1_even      |   106428 |
 Redistribution completed    |                   30 |                    30 | 2024-01-05
 02:24:20.584703 |         30 |         | Rebalance Disteven Table
 dev           | schemaefd028a2a48a4c | customer_even        |   130512 |
 Redistribution started      |                    0 |                       | 2024-01-05
 04:54:55.641741 |        190 |         | Restore Disteven Table
 dev           | schemaefd028a2a48a4c | customer_even        |   130512 |
 Redistribution checkpointed |     29.4342113157737 |     29.4342113157737 | 2024-01-05
 04:55:04.770696 |        190 |         | Restore Disteven Table
(8 rows)
```

SYS_RESTORE_STATE

使用 SYS_RESTORE_STATE 可以在傳統調整大小時監視每個資料表的移轉進度。當目標節點類型為 RA3 時，這特別適用。如需有關傳統調整為 RA3 節點大小的詳細資訊，請參閱傳統調整大小。

只有超級使用者才能看到 SYS_RESTORE_STATE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。

欄名稱	資料類型	描述
database_name	char(64)	資料庫中的資料表名稱。
schema_id	integer	資料表的結構描述 ID。
table_id	integer	資料表的 ID。
table_name	char(128)	資料表的名稱。
redistribution_status	char(128)	資料表重新分佈進度的狀態。可能值為 Completed 、In progress 及 Pending。
percentage_redistributed	float	資料表重新分佈進度的百分比。可能值為 0 到 100%。例如，值 25 表示 25% 的資料已重新分配。
redistribution_type	char(32)	資料表的重新發佈類型。KEY 轉換或 EVEN 重新平衡任務。如需分佈樣式的詳細資訊，請參閱分佈樣式。

範例查詢

下列查詢會傳回執行和排入佇列查詢的記錄。

```
SELECT * FROM sys_restore_state;
```

輸出範例。

```
 userid | database_name | schema_id | table_id |   table_name   | redistribution_status
 | precentage_redistributed |   redistribution_type
--------+---------------+-----------+----------+----------------
+----------------------+--------------------------+-------------------------
    1   |     test1     |   124865  |  124878  | customer_key_4 |         Pending
 |      0                   |  Rebalance Disteven Table
    1   |      dev      |   124865  |  124874  | customer_key_3 |         Pending
 |      0                   |  Rebalance Disteven Table
```

```
    1   |      dev      |   124865  |  124870  | customer_key_2 |        Completed
 |     100                |  Rebalance Disteven Table
    1   |      dev      |   124865  |  124866  | customer_key_1 |       In progress
 |     13.52              |  Restore Distkey Table
```

下面給你的數據處理狀態。

```
SELECT
    redistribution_status, ROUND(SUM(block_count) / 1024.0, 2) AS total_size_gb
FROM sys_restore_state sys inner join stv_tbl_perm stv
    on sys.table_id = stv.id
GROUP BY sys.redistribution_status;
```

輸出範例。

```
 redistribution_status | total_size_gb
-----------------------+---------------
 Completed             |          0.07
 Pending               |          0.71
 In progress           |          0.20
(3 rows)
```

SYS_SCHEMA_QUOTA_VIOLATIONS

超過結構描述配額時，記錄出現次數、交易 ID 和其他有用的資訊。此系統資料表為 STL_SCHEMA_QUOTA_VIOLATIONS 的翻譯。

所有使用者都可看見 r_SYS_SCHEMA_QUOTA_VIOLATIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
owner_id	integer	結構描述擁有者的 ID。
user_id	integer	產生項目的使用者之 ID。
transaction_id	bigint	與陳述式關聯的交易 ID。

欄名稱	資料類型	描述
session_id	integer	與陳述式相關聯的處理程序 ID。
schema_id	integer	命名空間或結構描述 ID。
schema_name	character (128)	命名空間或結構描述名稱。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。
record_time	沒有時區的時間戳記	發生違規的時間。

範例查詢

下列查詢顯示配額違規的結果：

```
SELECT user_id, TRIM(schema_name) "schema_name", quota, disk_usage, record_time FROM
sys_schema_quota_violations WHERE SCHEMA_NAME = 'sales_schema' ORDER BY timestamp DESC;
```

此查詢會傳回所指定結構描述的下列範例輸出：

```
user_id| schema_name  | quota | disk_usage | record_time
-------+--------------+-------+------------+----------------------------
104    | sales_schema | 2048  | 2798       | 2020-04-20 20:09:25.494723
(1 row)
```

SYS_SERVERLESS_USAGE

使用 SYS_SERVERLESS_USAGE 來檢視 Amazon Redshift Serverless 資源使用情況的詳細資訊。此系統檢視不適用於已佈建的 Amazon Redshift 叢集。

此檢視包含無伺服器使用量摘要，包括使用多少運算容量來處理查詢，以及使用的 Amazon Redshift 託管儲存量 (1 分鐘精細程度)。運算容量以 Redshift 處理單元 (RPU) 為單位進行測量，並以每秒 RPU 秒為單位執行的工作負載進行計量。RPU 用於處理對資料倉儲中載入的資料的查詢、從

Amazon S3 資料湖查詢或使用聯合查詢從操作資料庫存取的資料。Amazon Redshift 伺服器會在 SYS_SERVERLESS_USAGE 中保留 7 天的資訊。

如需運算成本帳單的範例，請參閱 Amazon Redshift Serverless 的計費。

只有超級使用者才能看到 SYS_SERVERLESS_USAGE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
start_time	timestamp	間隔開始的時間。
end_time	timestamp	間隔完成的時間。
compute_seconds	double precision	此時間間隔內耗用的累計運算單位 (RPU) 秒數。此值會計算為帳戶配置的基本 RPU 容量。
compute_capacity	double precision	在此時間間隔內配置的運算單元 (Redshift 處理單元或 RPU) 的平均數目。 Compute_capacity 值可以動態變更。
data_storage	integer	此時間間隔內使用的平均資料儲存空間 (MB)。 當從資料庫載入或刪除資料時，使用的資料儲存可以動態變化。
cross_region_transferred_data	integer	此時間間隔內，跨區域資料共用所傳輸的累計資料 (以位元組為單位)。
charged_seconds	integer	在此時間間隔內計費的累計運算單位 (RPU) 秒數。這

欄名稱	資料類型	描述
		是交易結束後計算，因此在交易執行時可以是 0。使用 charged_seconds 來計算 Amazon Redshift Serverless 工作群組的成本。此值將配置給 Amazon Redshift Serverless 工作群組的 RPU 容量納入考量。

使用須知

- 在某些情況下，compute_seconds 為 0，但 charged_seconds 大於 0，反之亦然。這是由於在系統檢視中記錄資料的方式所產生的正常行為。若要更準確地呈現無伺服器使用情況詳細資訊，我們建議您彙總資料。

範例

若要透過查詢 charged_seconds 來取得某個時間間隔內使用的 RPU 小時總費用，請執行下列查詢：

```
select trunc(start_time) "Day",
(sum(charged_seconds)/3600::double precision) * <Price for 1 RPU> as cost_incurred
from sys_serverless_usage
group by 1
order by 1
```

請注意，間隔期間可能會有閒置時間。閒置時間不會增加至使用的 RPU。

SYS_SESSION_HISTORY

使用 SYS_SESSION_HISTORY 來檢視目前作用中工作階段和工作階段歷史記錄的相關資訊。

所有使用者都可看見 SYS_SESSION_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	char(50)	產生項目之使用者的識別碼。
session_id	integer	與陳述式相關聯之工作階段的名稱。
database_name	char(50)	資料庫的名稱。
status	char	工作階段的狀態。可能值為 `active`、`timed out` 及 `closed`。
session_timeout	integer	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。
start_time	timestamp	建立連線的時間戳記。
end_time	timestamp	連線停止的時間戳記。

範例

下列為 SYS_SESSION_HISTORY 的範例輸出。

```
select * from sys_session_history;
 user_id | session_id |   database_name  | status | session_timeout |
 start_time         |          end_time
---------+------------+------------------+--------+-----------------
+----------------------------+----------------------------
       1 | 1073971370 | dev              | closed |        0        | 2023-07-17
 15:50:12.030104 | 2023-07-17 15:50:12.123218
       1 | 1073979694 | dev              | closed |        0        | 2023-07-17
 15:50:24.117947 | 2023-07-17 15:50:24.131859
       1 | 1073873049 | dev              | closed |        0        | 2023-07-17
 15:49:29.067398 | 2023-07-17 15:49:29.070294
       1 | 1073873086 | database18127a4a | closed |        0        | 2023-07-17
 15:49:29.119018 | 2023-07-17 15:49:29.125925
       1 | 1073832112 | dev              | closed |        0        | 2023-07-17
 15:49:29.164688 | 2023-07-17 15:49:29.179631
```

```
      1 | 1073987697 | dev              | closed |        0        | 2023-07-17
15:49:29.26749  | 2023-07-17 15:49:29.273034
      1 | 1073922429 | dev              | closed |        0        | 2023-07-17
15:49:33.35315  | 2023-07-17 15:49:33.367499
      1 | 1073766783 | dev              | closed |        0        | 2023-07-17
15:49:45.38237  | 2023-07-17 15:49:45.396902
      1 | 1073807506 | dev              | active |        0        | 2023-07-17
15:51:48        |
```

SYS_SPATIAL_SIMPLIFY

您可以使用 COPY 命令查詢系統檢視 SYS_SPATIAL_SIMPLIFY，以取得有關簡化空間幾何物件的資訊。當您在 Shapefile 上使用 COPY 時，您可以指定 SIMPLIFY `tolerance`、SIMPLIFY AUTO 和 SIMPLIFY AUTO `max_tolerance` 擷取選項。簡化的結果摘要在 SYS_SPATIAL_SIMPLIFY 系統檢視中。

設定 SIMPLIFY AUTO `max_tolerance` 時，此檢視會針對超出大小上限的每個幾何包含一列。設定 SIMPLIFY `tolerance` 時，會儲存整個 COPY 操作的一個列。此列參考 COPY 查詢 ID 和指定的簡化公差。

如需有關載入 shapefile 的相關資訊，請參閱將 Shapefile 載入 Amazon Redshift。

所有使用者都可看見 SYS_SPATIAL_SIMPLIFY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
query_id	bigint	產生此列的查詢 ID (COPY 命令)。
line_number	bigint	當指定 COPY `SIMPLIFY AUTO` 選項時，這個值是 Shapefile 中簡化記錄的記錄編號。
maximum_tolerance	double precision	COPY 命令中指定的距離公差值。這是使用 `SIMPLIFY AUTO` 選項的最大公差值，或使用 `SIMPLIFY` 選項的固定公差值。
initial_size	bigint	簡化之前 `GEOMETRY` 資料值的大小 (以位元組為單位)。

欄名稱	資料類型	描述
簡化	char(1)	指定 COPY SIMPLIFY AUTO 選項時，若已成功簡化幾何則為 t，否則為 f。如果使用給定的最大公差進行簡化後，幾何的大小仍大於幾何大小上限，則幾何可能無法成功簡化。
final_size	bigint	指定 COPY SIMPLIFY AUTO 選項時，這是簡化後幾何的位元組大小。
final_tolerance	double precision	為簡化選擇的最終公差。

範例查詢

下列查詢會傳回 COPY 簡化的記錄清單。

```
SELECT * FROM sys_spatial_simplify;


 query_id | line_number | maximum_tolerance | initial_size | simplified | final_size |
  final_tolerance
----------+-------------+-------------------+--------------+------------+------------
+-----------------------
    20    |     1184704 |                -1 |      1513736 | t          |    1008808 |
  1.276386653895e-05
    20    |     1664115 |                -1 |      1233456 | t          |    1023584 |
 6.11707814796635e-06
```

SYS_STREAM_SCAN_ERRORS

記錄透過串流擷取載入之記錄的錯誤。

所有使用者都可看見 SYS_STREAM_SCAN_ERRORS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
external_schema_name	character (128)	Kinesis 串流或 Amazon MSK 主題結構描述的名稱。區分大小寫。
stream_name	character (255)	串流或主題的名稱。區分大小寫。
mv_name	character (128)	相關聯具體化視觀表的名稱。如果沒有，則為空。區分大小寫。
transaction_id	bigint	交易 ID。
query_id	bigint	查詢 ID。
stream_timestamp_type	character (1)	串流時間戳記的類型。區分大小寫。
stream_timestamp	沒有時區的時間戳記	記錄到達的時間。
record_time	沒有時區的時間戳記	記錄錯誤訊息的時間。
partition_id	character (128)	分割區/碎片 ID。區分大小寫。
position	character (128)	記錄的位置。這與 Kinesis 中的序列號或 Amazon MSK 中的偏移量相對應。區分大小寫。
error_code	integer	錯誤代碼。

欄名稱	資料類型	描述
error_reason	character (128)	錯誤原因。區分大小寫。

SYS_STREAM_SCAN_STATES

記錄透過串流擷取載入之記錄的掃描狀態。

所有使用者都可看見 SYS_STREAM_SCAN_STATES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
external_schema_name	character (128)	外部結構描述名稱。區分大小寫。
stream_name	character (255)	串流名稱。區分大小寫。
mv_name	character (128)	相關聯具體化視觀表的名稱。如果沒有，則為空。區分大小寫。
transaction_id	bigint	交易 ID。
query_id	bigint	查詢 ID。
record_time	沒有時區的時間戳記	記錄資料的時間。

欄名稱	資料類型	描述
partition_id	character (128)	分割區或碎片 ID。區分大小寫。
latest_position	character (128)	批次中最後讀取的記錄的位置。這與 Kinesis 中的序列號或 Amazon MSK 中的偏移量相對應。區分大小寫。
scanned_rows	bigint	批次中掃描的記錄數。
skipped_rows	bigint	批次中略過的記錄數。
scanned_bytes	bigint	批次中掃描的位元組數。
stream_record_time_min	沒有時區的時間戳記	批次中最早記錄的 Kinesis 或 Amazon MSK 到達時間。
stream_record_time_max	沒有時區的時間戳記	批次中最新記錄的 Kinesis 或 Amazon MSK 到達時間。

下列查詢會顯示特定查詢的串流和主題資料。

```
select
 query_id,mv_name::varchar,external_schema_name::varchar,stream_name::varchar,sum(scanned_rows)
 total_records,
sum(scanned_bytes) total_bytes from sys_stream_scan_states where query in
 (5401180,8601939) group by 1,2,3,4;

  query_id  |    mv_name     | external_schema_name |   stream_name   | total_records |
  total_bytes
------------+----------------+----------------------+-----------------+---------------
+----------------
 5401180    | kinesistest    | kinesis              | kinesisstream   |    1493255696 |
 3209006490704
```

```
 8601939    | msktest        | msk                  | mskstream       |      14677023 |
   31056580668
(2 rows)
```

SYS_TRANSACTION_HISTORY

追蹤查詢時，請使用 SYS_TRANSACTION_HISTORY 來查看交易的詳細資訊。

只有超級使用者才能看到 SYS_TRANSACTION_HISTORY。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
transaction_id	bigint	交易的 ID。
isolation_level	text	交易的隔離層級。可能值為 `Serializable` 和 `Snapshot Isolation` 。
status	text	交易的狀態。可能的狀態為 `committed` 和 `rolledback` 。
transaction_start_time	timestamp	交易的開始時間。
commit_start_time	timestamp	遞交的開始時間。
commit_end_time	timestamp	提交的結束時間。
blocks_committed	bigint	必須寫入為此遞交一部分的區塊數目。
undo_transaction_id	bigint	復原交易的 ID (如果有任何交易已復原)。否則，此值為 -1。

範例查詢

```
select * from sys_transaction_history order by transaction_start_time desc;

 user_id | transaction_id | isolation_level |   status   |   transaction_start_time
   |     commit_start_time      |      commit_end_time       | blocks_committed |
 undo_transaction_id
---------+----------------+-----------------+------------+--------------------------
+---------------------------+---------------------------+------------------
+---------------------
     100 |           1310 | Serializable    | committed  | 2023-08-27 21:03:11.822205 |
 2023-08-28 21:03:11.825287 | 2023-08-28 21:03:11.854883 |               17 |
       -1
     101 |           1345 | Serializable    | committed  | 2023-08-27 21:03:12.000278 |
 2023-08-28 21:03:12.003736 | 2023-08-28 21:03:12.030061 |               17 |
       -1
     102 |           1367 | Serializable    | committed  | 2023-08-27 21:03:12.1532   |
 2023-08-28 21:03:12.156124 | 2023-08-28 21:03:12.186468 |               17 |
       -1
     100 |           1370 | Serializable    | committed  | 2023-08-27 21:03:12.199316 |
 2023-08-28 21:03:12.204854 | 2023-08-28 21:03:12.238186 |               24 |
       -1
     100 |           1408 | Serializable    | committed  | 2023-08-27 21:03:53.891107 |
 2023-08-28 21:03:53.894825 | 2023-08-28 21:03:53.927465 |               17 |
       -1
     100 |           1409 | Serializable    | rolledback | 2023-08-27 21:03:53.936431 |
 2000-01-01 00:00:00        | 2023-08-28 21:04:08.712532 |                0 |
      1409
     101 |           1415 | Serializable    | committed  | 2023-08-27 21:04:24.283188 |
 2023-08-28 21:04:24.289196 | 2023-08-28 21:04:24.374318 |               25 |
       -1
     101 |           1416 | Serializable    | committed  | 2023-08-27 21:04:24.38818  |
 2023-08-28 21:04:24.391688 | 2023-08-28 21:04:24.415135 |               17 |
       -1
     100 |           1417 | Serializable    | rolledback | 2023-08-27 21:04:24.424252 |
 2000-01-01 00:00:00        | 2023-08-28 21:04:28.354826 |                0 |
      1417
     101 |           1418 | Serializable    | rolledback | 2023-08-27 21:04:24.425195 |
 2000-01-01 00:00:00        | 2023-08-28 21:04:28.680355 |                0 |
      1418
     100 |           1420 | Serializable    | committed  | 2023-08-27 21:04:28.697607 |
 2023-08-28 21:04:28.702374 | 2023-08-28 21:04:28.735541 |               23 |
       -1
```

```
    101 |           1421 | Serializable    | committed  | 2023-08-27 21:04:28.744854 |
2023-08-28 21:04:28.749344 | 2023-08-28 21:04:28.779029 |               23 |
       -1
    101 |           1423 | Serializable    | committed  | 2023-08-27 21:04:28.78942  |
2023-08-28 21:04:28.791556 | 2023-08-28 21:04:28.817485 |               16 |
       -1
    100 |           1430 | Serializable    | committed  | 2023-08-27 21:04:28.917788 |
2023-08-28 21:04:28.919993 | 2023-08-28 21:04:28.944812 |               16 |
       -1
    102 |           1494 | Serializable    | committed  | 2023-08-27 21:04:37.029058 |
2023-08-28 21:04:37.033137 | 2023-08-28 21:04:37.062001 |               16 |
       -1
```

SYS_UDF_LOG

使用者定義函數 (UDF) 執行期間產生的記錄系統定義的錯誤和警告訊息。

只有超級使用者才能看到 SYS_UDF_LOG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
query_id	bigint	查詢識別碼。
function_name	text	使用者定義函數的名稱。
record_time	timestamp	建立記錄的時間。
sequence	integer	單一記錄訊息的順序。
message	text	記錄訊息文字。

範例查詢

以下範例說明 UDF 如何處理系統定義的錯誤。第一個區塊顯示傳回引數反向之 UDF 函數的定義。當您執行函數並提供 0 做為引數時，函數會傳回錯誤。最後一個陳述式會傳回 SYS_UDF_LOG 中記錄的錯誤訊息。

```
-- Create a function to find the inverse of a number.
CREATE OR REPLACE FUNCTION f_udf_inv(a int)

RETURNS float

IMMUTABLE AS $$return 1/a

$$ LANGUAGE plpythonu;

-- Run the function with 0 to create an error.
Select f_udf_inv(0);

-- Query SYS_UDF_LOG to view the message.
Select query_id, record_time, message::varchar from sys_udf_log;


query_id    |    record_time              |                 message
----------+----------------------------
+-----------------------------------------------------------
2211        | 2023-08-23 15:53:11.360538 |  ZeroDivisionError: integer division or
 modulo by zero line 2, in f_udf_inv\n return 1/a\n
```

下列範例會將記錄且警告訊息新增至 UDF，以至於除以零操作會導致警告訊息，而不是停止並出現錯誤訊息。

```
-- Create a function to find the inverse of a number and log a warning if you input 0.
CREATE OR REPLACE FUNCTION f_udf_inv_log(a int)
  RETURNS float IMMUTABLE
 AS $$
  import logging
  logger = logging.getLogger() #get root logger
  if a==0:
    logger.warning('You attempted to divide by zero.\nReturning zero instead of error.
\n')
    return 0
  else:
     return 1/a
$$ LANGUAGE plpythonu;

-- Run the function with 0 to trigger the warning.
Select f_udf_inv_log(0);

-- Query SYS_UDF_LOG to view the message.
```

```
Select query_id, record_time, message::varchar from sys_udf_log;

 query_id |        record_time         |                                        message
----------+----------------------------
+-------------------------------------------------------------------------------
     0   | 2023-08-23 16:10:48.833503 | WARNING: You attempted to divide by zero.
\nReturning zero instead of error.\n
```

SYS_UNLOAD_DETAIL

使用 SYS_UNLOAD_DETAIL 來檢視 UNLOAD 操作的詳細資訊。它會針對 UNLOAD 陳述式所建立的每一個檔案各記錄一列。例如，若 UNLOAD 建立 12 個檔案，則 SYS_UNLOAD_DETAIL 將包含 12 個對應列。

所有使用者都可看見 SYS_UNLOAD_DETAIL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目之使用者的識別碼。
query_id	integer	UNLOAD 命令的查詢識別碼。
session_id	integer	與查詢陳述式相關聯的處理程序 ID。
transaction_id	bigint	與查詢陳述式關聯的交易 ID。
file_name	character (1280)	檔案的完整 Amazon S3 物件路徑。
start_time	timestamp	交易開始的時間。
end_time	timestamp	交易完成的時間。
line_count	bigint	已卸載至檔案的行數 (列數)。
transfer_size	bigint	已傳輸的位元組數目。

欄名稱	資料類型	描述
file_format	character (10)	卸載檔案的檔案格式。

範例查詢

下列查詢顯示卸載查詢詳細資訊，包括卸載命令的格式、列和檔案計數。

```
select query_id, substring(file_name, 0, 50), transfer_size, file_format from
 sys_unload_detail;
```

輸出範例。

```
 query_id |                    substring                     | transfer_size |
 file_format
----------+--------------------------------------------------+---------------
+-------------
     9272 | s3://my-bucket/my_unload_doc_venue0000_part_00.gz |        395886 | Text

     9272 | s3://my-bucket/my_unload_doc_venue0001_part_00.gz |        406444 | Text

     9272 | s3://my-bucket/my_unload_doc_venue0002_part_00.gz |        409431 | Text

     9272 | s3://my-bucket/my_unload_doc_venue0003_part_00.gz |        403051 | Text

     9272 | s3://my-bucket/my_unload_doc_venue0004_part_00.gz |        413592 | Text

     9272 | s3://my-bucket/my_unload_doc_venue0005_part_00.gz |        395689 | Text

(6 rows)
```

SYS_UNLOAD_HISTORY

使用 SYS_UNLOAD_HISTORY 來檢視 UNLOAD 命令的詳細資訊。每一列代表一個 UNLOAD 命令，其中包含某些欄位的累計統計資料。它包含正在執行和已完成的 UNLOAD 命令。

所有使用者都可看見 SYS_UNLOAD_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交卸載之使用者的識別碼。
query_id	bigint	UNLOAD 命令的查詢識別碼。
transaction_id	bigint	交易識別碼。
session_id	integer	執行卸載之程序的處理程序識別碼。
database_name	text	發出操作時，要將使用者連接至其中的資料庫名稱。
status	text	UNLOAD 命令的狀態。有效值包括：`running`、`completed`、`aborted` 與 `unknown`。
start_time	timestamp	開始卸載的時間。
end_time	timestamp	卸載完成的時間。
duration	bigint	UNLOAD 命令花費的時間 (微秒)。
file_format	text	輸出檔案的檔案格式。
compression_type	text	壓縮類型。
unloaded_location	text	已卸載檔案的 Amazon S3 位置。
unloaded_rows	bigint	列的數。
unloaded_files_count	bigint	輸出檔案的檔案計數。
unloaded_files_size	bigint	輸出檔案的檔案大小。
error_message	text	UNLOAD 命令的錯誤訊息。

範例查詢

下列查詢顯示卸載查詢詳細資訊，包括卸載命令的格式、列和檔案計數。

```
SELECT query_id,
       file_format,
       start_time,
       duration,
       unloaded_rows,
       unloaded_files_count
FROM sys_unload_history
ORDER BY query_id,
file_format limit 100;
```

輸出範例。

```
 query_id | file_format |         start_time         | duration | unloaded_rows |
 unloaded_files_count
----------+-------------+----------------------------+----------+---------------
+----------------------
   527067 | Text        | 2022-02-09 05:18:35.844452 |  5932478 |            10 |
              1
```

SYS_USERLOG

記錄資料庫使用者之下列變更的詳細資訊：

- 建立使用者
- 捨棄使用者
- 更改使用者 (重新命名)
- 更改使用者 (更改屬性)

您可以查詢此檢視，以查看有關無伺服器工作群組和已佈建叢集的資訊。

只有超級使用者才能看到 SYS_USERLOG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交卸載之使用者的識別碼。
user_name	character(50)	受到變更影響之使用者的使用者名稱。
original_user_name	character(50)	重新命名動作中的原始使用者名稱。若是任何其他動作，此欄位空白。
動作	character(10)	發生的動作。有效值包括 alter、create、drop 和 rename。
has_create_db_privs	integer	如果為 true (值為 1)，則使用者具有建立資料庫許可。
is_superuser	integer	若為 true (值為 1)，使用者可以更新系統目錄。
has_update_catalog_privs	integer	若為 true (值為 1)，使用者可以更新系統目錄。
password_expiration	timestamp	密碼到期日。
session_id	integer	處理程序 ID。
transaction_id	bigint	交易 ID。
record_time	timestamp	查詢開始的時間，以 UTC 表示。

範例查詢

下列範例會執行四個使用者動作，然後查詢 SYS_USERLOG 資料表。

```
CREATE USER userlog1 password 'Userlog1';
ALTER USER userlog1 createdb createuser;
ALTER USER userlog1 rename  to userlog2;
DROP user userlog2;

SELECT user_id, user_name, original_user_name, action, has_create_db_privs,
 is_superuser from SYS_USERLOG order by record_time desc;
```

```
user_id |  user_name | original_user_name |  action | has_create_db_privs |
 is_superuser
--------+------------+--------------------+---------+---------------------+------------
    108 |  userlog2  |                    | drop    |                   1 |   1
    108 |  userlog2  |     userlog1       | rename  |                   1 |   1
    108 |  userlog1  |                    | alter   |                   1 |   1
    108 |  userlog1  |                    | create  |                   0 |   0
 (4 rows)
```

SYS_VACUUM_HISTORY

使用 SYS_VACUUM_HISTORY 來檢視清空查詢的詳細資料。如需 VACUUM 命令的詳細資訊，請參閱VACUUM。

所有使用者都可看見 SYS_VACUUM_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
user_id	integer	啟動查詢的使用者 ID。
transaction_id	long	VACUUM 陳述式的交易 ID。
query_id	long	VACUUM 陳述式的查詢識別碼。您可以將此資料表聯結至 SYS_QUERY_DETAIL 檢視，以查看針對特定 VACUUM 交易執行的個別 SQL 陳述式。如

欄名稱	資料類型	描述
		果清空整個資料庫，則會清空個別交易中的每個資料表。對於自動化 VACUUM 操作，此值為 null。
database_name	text	資料庫的名稱。
schema_name	text	結構描述的名稱。
table_name	text	資料表的名稱。
table_id	integer	資料表的 ID。
vacuum_type	character	VACUUM 操作的類型。可能的值如下： • **Delete** • **Sort** • **Reindex** • **Recluster** • **Full** 如需清空類型的相關資訊，請參閱VACUUM。
is_automatic	boolean	如果操作是自動清空則為 `true`。否則為 `false`。

欄名稱	資料類型	描述
status	character	描述在清空操作過程中所做的目前活動： • 初始化 • Sort • Merge • Delete • Select • 失敗 • 完成 • 略過 • 建置 INTERLEAVED SORTKEY 順序
start_time	timestamp	清空操作開始的時間。
end_time	timestamp	清空操作結束的時間。如果操作正在進行中，則此欄位為空白。
record_time	timestamp	清空操作記錄在 SYS_VACUUM_HISTORY 中的時間。
持續時間	integer	清空操作開始和結束之間的微秒數。如果清空操作正在進行中，則此欄位為空白。
rows_before_vacuum	bigint	資料表中的實際資料列數目加上任何仍在磁碟上儲存的已刪除資料列 (等待清空)。
size_before_vacuum	integer	清空操作開始之前的資料表大小，以 MB 為單位。

欄名稱	資料類型	描述
reclaimable_rows	bigint	清空操作在開始之前估計將回收的列數。
reclaimed_rows	bigint	清空操作回收的列數。
reclaimed_blocks	bigint	清空操作回收的區塊數。
sortedrows_before_vacuum	integer	清空操作開始之前資料表中已排序的列數。
sortedrows_after_vacuum	integer	清空操作完成後，資料表中額外排序的列數。這不包括計數的行`sortedrows_before_vacuum` 。

用於移轉至 SYS 監視檢視的系統檢視對映

將 Amazon Redshift 佈建叢集遷移到 Amazon Redshift Serverless 時，您的監控或診斷查詢可能會參考僅在已佈建叢集上可用的系統檢視。您可以更新查詢以使用 SYS 監控檢視。此頁面提供僅啟動設定至 SYS 視觀表對應，供您在更新查詢時參照。

主題

- SYS_QUERY_HISTORY
- SYS_QUERY_DETAIL
- SYS_RESTORE_LOG
- SYS_RESTORE_STATE
- SYS_TRANSACTION_HISTORY
- SYS_QUERY_TEXT
- SYS_CONNECTION_LOG
- SYS_SESSION_HISTORY
- SYS_LOAD_DETAIL
- SYS_LOAD_HISTORY
- SYS_LOAD_ERROR_DETAIL

- SYS_UNLOAD_HISTORY
- SYS_UNLOAD_DETAIL
- SYS_COPY_REPLACEMENTS
- SYS_DATASHARE_USAGE_CONSUMER
- SYS_DATASHARE_USAGE_PRODUCER
- SYS_DATASHARE_CROSS_REGION_USAGE
- SYS_DATASHARE_CHANGE_LOG
- SYS_EXTERNAL_QUERY_DETAIL
- SYS_EXTERNAL_QUERY_ERROR
- SYS_VACUUM_HISTORY
- SYS_ANALYZE_HISTORY
- SYS_ANALYZE_COMPRESSION_HISTORY
- SYS_MV_REFRESH_HISTORY
- SYS_MV_STATE
- SYS_PROCEDURE_CALL
- SYS_PROCEDURE_MESSAGES
- SYS_UDF_LOG
- SYS_USERLOG
- SYS_SCHEMA_QUOTA_VIOLATIONS
- SYS_SPATIAL_SIMPLIFY

SYS_QUERY_HISTORY

下列資料表中的部分或所有欄也會在 SYS_QUERY_HISTORY中定義。

- STL_DDLTEXT
- STL_ERROR
- STL_QUERY
- STL_UTILITYTEXT
- STL_WLM_QUERY
- STV_INFLIGHT

- STV_RECENTS
- STV_WLM_QUERY_STATE
- SVL_COMPILE
- SVL_MULTI_STATEMENT_VIOLATIONS
- SVL_QLOG
- SVL_QUERY_QUEUE_INFO
- SVL_STATEMENTTEXT
- SVL_TERMINATE

SYS_QUERY_DETAIL

下列資料表中的部分或所有欄也會在 SYS_QUERY_DETAIL中定義。

- STL_AGGR
- STL_ALERT_EVENT_LOG
- STL_BCAST
- STL_DELETE
- STL_DIST
- STL_EXPLAIN
- STL_HASH
- STL_HASHJOIN
- STL_INSERT
- STL_LIMIT
- STL_MERGE
- STL_MERGEJOIN
- STL_NESTLOOP
- STL_PARSE
- STL_PLAN_INFO
- STL_PROJECT
- STL_QUERY_METRICS
- STL_RETURN

- STL_SAVE
- STL_SCAN
- STL_SORT
- STL_STREAM_SEGS
- STL_UNIQUE
- STL_WINDOW
- STV_EXEC_STATE
- STV_QUERY_METRICS
- SVCS_QUERY_SUMMARY
- SVL_QUERY_METRICS
- SVL_QUERY_METRICS_SUMMARY
- SVL_QUERY_REPORT
- SVL_QUERY_SUMMARY
- SVV_QUERY_STATE

SYS_RESTORE_LOG

下列資料表中的部分或所有欄也會在 SYS_RESTORE_LOG中定義。

- SVL 還原 _ 更改表 _ 進度

SYS_RESTORE_STATE

下列資料表中的部分或所有欄也會在 SYS_RESTORE_STATE中定義。

- STV_X 還原 _ 交換佇列狀態

SYS_TRANSACTION_HISTORY

下列資料表中的部分或所有欄也會在 SYS_TRANSACTION_HISTORY中定義。

- STL_COMMIT_STATS
- STL_TR_CONFLICT

- STL_UNDONE

SYS_QUERY_TEXT

下列資料表中的部分或所有欄也會在 SYS_QUERY_TEXT中定義。

- STL_QUERYTEXT

SYS_CONNECTION_LOG

下列資料表中的部分或所有欄也會在 SYS_CONNECTION_LOG中定義。

- STL_CONNECTION_LOG

SYS_SESSION_HISTORY

下列資料表中的部分或所有欄也會在 SYS_SESSION_HISTORY中定義。

- STL_SESSIONS
- STL_RESTARTED_SESSIONS
- STV_SESSIONS

SYS_LOAD_DETAIL

下列資料表中的部分或所有欄也會在 SYS_LOAD_DETAIL中定義。

- STL_LOAD_COMMITS

SYS_LOAD_HISTORY

下列資料表中的部分或所有欄也會在 SYS_LOAD_HISTORY中定義。

- STL_LOAD_COMMITS

SYS_LOAD_ERROR_DETAIL

下列資料表中的部分或所有欄也會在 SYS_LOAD_ERROR_DETAIL中定義。

- STL_LOADERROR_DETAIL
- STL_LOAD_ERRORS

SYS_UNLOAD_HISTORY

下列資料表中的部分或所有欄也會在 SYS_UNLOAD_HISTORY中定義。

- STL_UNLOAD_LOG

SYS_UNLOAD_DETAIL

下列資料表中的部分或所有欄也會在 SYS_UNLOAD_DETAIL中定義。

- STL_UNLOAD_LOG

SYS_COPY_REPLACEMENTS

下列資料表中的部分或所有欄也會在 SYS_COPY_REPLACEMENTS中定義。

- STL_REPLACEMENTS

SYS_DATASHARE_USAGE_CONSUMER

下列資料表中的部分或所有欄也會在 SYS_DATASHARE_USAGE_CONSUMER中定義。

- SVL_DATASHARE_USAGE_CONSUMER

SYS_DATASHARE_USAGE_PRODUCER

下列資料表中的部分或所有欄也會在 SYS_DATASHARE_USAGE_PRODUCER中定義。

- SVL_DATASHARE_USAGE_PRODUCER

SYS_DATASHARE_CROSS_REGION_USAGE

下列資料表中的部分或所有欄也會在 SYS_DATASHARE_CROSS_REGION_USAGE中定義。

- SVL_DATASHARE_CROSS_REGION_USAGE

SYS_DATASHARE_CHANGE_LOG

下列資料表中的部分或所有欄也會在 SYS_DATASHARE_CHANGE_LOG中定義。

- SVL_DATASHARE_CHANGE_LOG

SYS_EXTERNAL_QUERY_DETAIL

下列資料表中的部分或所有欄也會在 SYS_EXTERNAL_QUERY_DETAIL中定義。

- SVL_FEDERATED_QUERY
- SVL_S3LIST
- SVL_S3QUERY
- SVL_S3QUERY_SUMMARY

SYS_EXTERNAL_QUERY_ERROR

下列資料表中的部分或所有欄也會在 SYS_EXTERNAL_QUERY_ERROR中定義。

- SVL_SPECTRUM_SCAN_ERROR

SYS_VACUUM_HISTORY

下列資料表中的部分或所有欄也會在 SYS_VACUUM_HISTORY中定義。

- STL_VACUUM
- SVL_VACUUM_PERCENTAGE
- SVV_VACUUM_PROGRESS
- SVV_VACUUM_SUMMARY

SYS_ANALYZE_HISTORY

下列資料表中的部分或所有欄也會在 SYS_ANALYZE_HISTORY中定義。

- STL_ANALYZE

SYS_ANALYZE_COMPRESSION_HISTORY

下列資料表中的部分或所有欄也會在 SYS_ANALYZE_COMPRESSION_HISTORY中定義。

- STL_ANALYZE_COMPRESSION

SYS_MV_REFRESH_HISTORY

下列資料表中的部分或所有欄也會在 SYS_MV_REFRESH_HISTORY中定義。

- SVL_MV_REFRESH_STATUS

SYS_MV_STATE

下列資料表中的部分或所有欄也會在 SYS_MV_STATE中定義。

- STL_MV_STATE

SYS_PROCEDURE_CALL

下列資料表中的部分或所有欄也會在 SYS_PROCEDURE_CALL中定義。

- SVL_STORED_PROC_CALL

SYS_PROCEDURE_MESSAGES

下列資料表中的部分或所有欄也會在 SYS_PROCEDURE_MESSAGES中定義。

- SVL_STORED_PROC_MESSAGES

SYS_UDF_LOG

下列資料表中的部分或所有欄也會在 SYS_UDF_LOG中定義。

- SVL_UDF_LOG

SYS_USERLOG

下列資料表中的部分或所有欄也會在 SYS_USERLOG中定義。

- STL_USERLOG

SYS_SCHEMA_QUOTA_VIOLATIONS

下列資料表中的部分或所有欄也會在 SYS_SCHEMA_QUOTA_VIOLATIONS中定義。

- STL_SCHEMA_QUOTA_VIOLATIONS

SYS_SPATIAL_SIMPLIFY

下列資料表中的部分或所有欄也會在 SYS_SPATIAL_SIMPLIFY中定義。

- SVL_SPATIAL_SIMPLIFY

系統監控 (僅限已佈建)

您可以在已佈建的叢集上查詢下列系統資料表和檢視。STL 和 STV 資料表和檢視包含在數個系統資料表中找到的資料子集。這些項目可讓您快速輕鬆存取在那些資料表中找到的常見查詢資料。

SVCS 檢視可提供主要叢集與並行擴展叢集查詢的詳細資訊。SVL 檢視僅針對在主叢集上執行的查詢提供資訊，但 SVL_STATEMENTTEXT 除外。SVL_STATMENTTEXT 可以包含在並行擴展叢集以及主叢集上執行之查詢的資訊。

主題

- 用於記錄日誌的 STL 檢視
- 快照資料的 STV 資料表
- 主要和並行擴展叢集的 SVCS 檢視
- 主要叢集的 SVL 檢視

用於記錄日誌的 STL 檢視

STL 系統檢視是從 Amazon Redshift 日誌檔產生，以提供系統歷史記錄。

這些檔案位於資料倉儲叢集中的每一個節點上。STL 檢視會從日誌中取得資訊，並將它們格式化為可供系統管理員使用的檢視。

日誌保留 - STL 系統檢視會保留 7 天的日誌歷史記錄。所有叢集大小和節點類型都保證日誌保留，並且不受叢集工作負載變化的影響。日誌保留也不受叢集狀態的影響，例如叢集暫停時。只有在叢集是新的情況下，您才會有少於 7 天的日誌歷史記錄。您不需要採取任何動作來保留日誌，但必須定期將日誌資料複製到其他表格或卸載到 Amazon S3，以保留 7 天以上的日誌資料。

主題

- STL_AGGR
- STL_ALERT_EVENT_LOG
- STL_ANALYZE
- STL_ANALYZE_COMPRESSION
- STL_BCAST
- STL_COMMIT_STATS
- STL_CONNECTION_LOG
- STL_DDLTEXT
- STL_DELETE
- STL_DISK_FULL_DIAG
- STL_DIST
- STL_ERROR
- STL_EXPLAIN
- STL_FILE_SCAN
- STL_HASH
- STL_HASHJOIN
- STL_INSERT
- STL_LIMIT
- STL_LOAD_COMMITS
- STL_LOAD_ERRORS
- STL_LOADERROR_DETAIL
- STL_MERGE
- STL_MERGEJOIN

- STL_MV_STATE
- STL_NESTLOOP
- STL_PARSE
- STL_PLAN_INFO
- STL_PROJECT
- STL_QUERY
- STL_QUERY_METRICS
- STL_QUERYTEXT
- STL_REPLACEMENTS
- STL_RESTARTED_SESSIONS
- STL_RETURN
- STL_S3CLIENT
- STL_S3CLIENT_ERROR
- STL_SAVE
- STL_SCAN
- STL_SCHEMA_QUOTA_VIOLATIONS
- STL_SESSIONS
- STL_SORT
- STL_SSHCLIENT_ERROR
- STL_STREAM_SEGS
- STL_TR_CONFLICT
- STL_UNDONE
- STL_UNIQUE
- STL_UNLOAD_LOG
- STL_USAGE_CONTROL
- STL_USERLOG
- STL_UTILITYTEXT
- STL_VACUUM
- STL_WINDOW
- STL_WLM_ERROR

- STL_WLM_RULE_ACTION
- STL_WLM_QUERY

STL_AGGR

分析查詢的彙總執行步驟。在執行彙總函數和 GROUP BY 子句期間，即會進行這些步驟。

所有使用者都可看見 STL_AGGR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_AGGR 僅包含在主要叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

欄名稱	資料類型	描述
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
slots	integer	雜湊儲存貯體的數目。
occupied	integer	包含記錄之插槽的數目。
maxlength	integer	最大插槽的大小。
tbl	integer	表格 ID。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體的位元組數。
type	character(6)	步驟的類型。有效的 值如下： • HASHED。表示步驟已使用分組且未排序的彙整。 • PLAIN。表示步驟已使用未分組的純量彙整。 • SORTED。表示步驟已使用分組且排序的彙整。
resizes	integer	此資訊僅供內部使用。
flushable	integer	此資訊僅供內部使用。

範例查詢

傳回 SLICE 1 和 TBL 239 之彙總執行步驟的相關資訊。

```
select query, segment, bytes, slots, occupied, maxlength, is_diskbased, workmem, type
from stl_aggr where slice=1 and tbl=239
order by rows
limit 10;
```

```
 query | segment | bytes |  slots  | occupied | maxlength | is_diskbased |  workmem  |
 type
-------+---------+-------+---------+----------+-----------+--------------+-----------
+--------
   562 |       1 |     0 | 4194304 |        0 |         0 | f            | 383385600 |
 HASHED
   616 |       1 |     0 | 4194304 |        0 |         0 | f            | 383385600 |
 HASHED
   546 |       1 |     0 | 4194304 |        0 |         0 | f            | 383385600 |
 HASHED
   547 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
   685 |       1 |    32 | 4194304 |        1 |         0 | f            | 383385600 |
 HASHED
   652 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
   680 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
   658 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
   686 |       0 |     8 |       0 |        0 |         0 | f            |         0 |
 PLAIN
   695 |       1 |    32 | 4194304 |        1 |         0 | f            | 383385600 |
 HASHED
(10 rows)
```

STL_ALERT_EVENT_LOG

當查詢最佳化器識別可能表示效能問題的狀況時，請記錄一個提醒。使用 STL_ALERT_EVENT_LOG 檢視來識別提升查詢效能的機會。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱 查詢處理。

所有使用者都可看見 STL_ALERT_EVENT_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_ALERT_EVENT_LOG 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
pid	integer	與陳述式和配量相關聯的處理程序 ID。如果相同的查詢在多個配量上執行，則其可能具有多個 PID。
xid	bigint	與陳述式關聯的交易 ID。
事件	character (1024)	提醒事件的說明。
solution	character (1024)	建議的解決方案。
event_time	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

使用須知

您可以使用 STL_ALERT_EVENT_LOG，識別查詢中的潛在問題，然後遵循調校查詢效能中的實務，來最佳化資料庫設計並重新撰寫查詢。STL_ALERT_EVENT_LOG 會記錄下列提醒：

- 找不到統計資訊

  找不到統計資訊。在進行資料載入或重要更新之後執行 ANALYZE，並使用 STATUPDATE 與 COPY 操作搭配。如需詳細資訊，請參閱 設計查詢的 Amazon Redshift 最佳實務。

- 巢狀迴圈

巢狀迴路通常是 Cartesian 產品。評估您的查詢，以確保所有參與資料表均已有效聯結。

- 選擇性相當高的篩選條件

  傳回的資料列與已掃描資料列的比率低於 0.05。已掃描列是 rows_pre_user_filter 的值，而傳回的列則是 STL_SCAN 系統檢視中的列值。表示查詢正在掃描例外狀況大量的資料列來決定結果集。這可能是由於找不到排序索引鍵或其不正確所致。如需詳細資訊，請參閱 使用排序索引鍵。

- 過多的幽靈資料列

  掃描已略過相當多標示為已刪除但未清空的資料列，或已插入但未遞交的資料列。如需詳細資訊，請參閱 清空資料表。

- 大型分佈

  已重新配送超過 1,000,000 個資料列，進行雜湊聯結或彙整。如需詳細資訊，請參閱 使用資料分佈樣式。

- 大型廣播

  已播送超過 1,000,000 個資料列，進行雜湊聯結。如需詳細資訊，請參閱 使用資料分佈樣式。

- 序列執行

  已在查詢計劃中指出 DS_DIST_ALL_INNER 重新配送樣式，其會強制序列執行，因為整個內部資料表已重新配送至單一節點。如需詳細資訊，請參閱 使用資料分佈樣式。

範例查詢

下列查詢顯示四個查詢的提醒事件。

```
SELECT query, substring(event,0,25) as event,
substring(solution,0,25) as solution,
trim(event_time) as event_time from stl_alert_event_log order by query;

 query |             event             |          solution            |     event_time

-------+-------------------------------+-------------------------------
+---------------------
  6567 | Missing query planner statist | Run the ANALYZE command       | 2014-01-03
 18:20:58
  7450 | Scanned a large number of del | Run the VACUUM command to rec| 2014-01-03
 21:19:31
```

```
  8406 | Nested Loop Join in the query | Review the join predicates to| 2014-01-04
 00:34:22
 29512 | Very selective query filter:r | Review the choice of sort key| 2014-01-06
 22:00:00

(4 rows)
```

STL_ANALYZE

記錄 ANALYZE 操作的詳細資訊。

只有超級使用者才能看到 STL_ANALYZE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_ANALYZE_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	long	交易 ID。
database	char(30)	資料庫名稱。
table_id	integer	資料表 ID。
status	char(15)	分析命令的結果。可能值為 `Full`、`Skipped` 及 `Predicate Column` 。
rows	double	資料表中的列總數。
modified_rows	double	自前次 ANALYZE 操作後已修改的資料列總數。
threshold_percent	integer	`analyze_threshold_percent` 參數的值。

欄名稱	資料類型	描述
is_auto	char(1)	如果操作預設包含 Amazon Redshift 分析操作，則值為 true (t)。如果明確執行 ANALYZE 命令，則值為 false (f)。
starttime	timestamp	分析操作開始執行的時間，以 UTC 表示。
endtime	timestamp	分析操作完成執行的時間，以 UTC 表示。
prevtime	timestamp	先前分析資料表的時間，以 UTC 表示。
num_predicate_cols	integer	資料表中述詞資料欄的目前數目。
num_new_predicate_cols	integer	自前一個分析操作後資料表中新述詞資料欄的數目。
is_background	character(1)	如果分析是由自動分析操作執行，則值為 true (t)。否則值為 false (f)。
auto_analyze_phase	character(100)	保留供內部使用。
schema_name	char(128)	資料表的結構描述名稱。
table_name	char(136)	資料表的名稱。

範例查詢

下列範例聯結 STV_TBL_PERM，以顯示資料表名稱和執行結果。

```
select distinct a.xid, trim(t.name) as name, a.status, a.rows, a.modified_rows,
 a.starttime, a.endtime
from stl_analyze a
join stv_tbl_perm t  on t.id=a.table_id
where name = 'users'
order by starttime;
```

```
xid    | name  | status          | rows  | modified_rows | starttime           |
 endtime
-------+-------+-----------------+-------+---------------+---------------------
+--------------------
  1582 | users | Full            | 49990 |         49990 | 2016-09-22 22:02:23 |
 2016-09-22 22:02:28
244287 | users | Full            | 24992 |         74988 | 2016-10-04 22:50:58 |
 2016-10-04 22:51:01
244712 | users | Full            | 49984 |         24992 | 2016-10-04 22:56:07 |
 2016-10-04 22:56:07
245071 | users | Skipped         | 49984 |             0 | 2016-10-04 22:58:17 |
 2016-10-04 22:58:17
245439 | users | Skipped         | 49984 |          1982 | 2016-10-04 23:00:13 |
 2016-10-04 23:00:13
(5 rows)
```

STL_ANALYZE_COMPRESSION

記錄在 COPY 或 ANALYZE COMPRESSION 命令期間的壓縮分析操作詳細資訊。

所有使用者都可看見 STL_ANALYZE_COMPRESSION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_ANALYZE_COMPRESSION_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
start_time	timestamp	壓縮分析操作開始進行的時間。
xid	bigint	壓縮分析操作的交易 ID。
tbl	integer	接受分析之資料表的 ID。
資料表名稱	character(128)	接受分析之資料表的名稱。
col	integer	接受分析之資料表的資料欄索引，用於判斷壓縮編碼。

欄名稱	資料類型	描述
old_encoding	character(15)	壓縮分析之後的編碼類型。
new_encoding	character(15)	壓縮分析之後的編碼類型。
模式	character(14)	可能值如下： PRESET 指定 new_encoding 是由 Amazon Redshift COPY 命令根據欄資料類型決定。不會對任何資料取樣。 ON 指定 new_encoding 是由 Amazon Redshift COPY 命令根據範例資料的分析決定。 ANALYZE ONLY 指定 new_encoding 是由 Amazon Redshift ANALYZE COMPRESSION 命令根據範例資料的分析決定。然而，不會變更已分析資料欄的編碼類型。
最佳壓縮 _ 編碼	character(15)	提供最佳壓縮率的編碼類型。
建議位元組	character(15)	採用新編碼所使用的位元組。
最佳壓縮 _ 位元組	character(15)	採用最佳壓縮編碼所使用的位元組。
NDV	bigint	取樣列中不同值的數目。

範例查詢

下列範例由在相同工作階段中執行的最後一項 COPY 命令檢查 lineitem 資料表上的壓縮分析詳細資料。

```
select xid, tbl, btrim(tablename) as tablename, col, old_encoding, new_encoding,
 best_compression_encoding, mode
from stl_analyze_compression
where xid = (select xid from stl_query where query = pg_last_copy_id()) order by col;

 xid  |  tbl   | tablename | col |  old_encoding   |  new_encoding   |
 best_compression_encoding |      mode
------+--------+-----------+-----+-----------------+-----------------
+---------------------------+----------------
 5308 | 158961 | $lineitem |   0 | mostly32        | az64            | delta
          | ON
 5308 | 158961 | $lineitem |   1 | mostly32        | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   2 | lzo             | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   3 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   4 | bytedict        | az64            | bytedict
          | ON
 5308 | 158961 | $lineitem |   5 | mostly32        | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   6 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   7 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |   8 | lzo             | lzo             | lzo
          | ON
 5308 | 158961 | $lineitem |   9 | runlength       | runlength       | runlength
          | ON
 5308 | 158961 | $lineitem |  10 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |  11 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |  12 | delta           | az64            | az64
          | ON
 5308 | 158961 | $lineitem |  13 | bytedict        | bytedict        | bytedict
          | ON
 5308 | 158961 | $lineitem |  14 | bytedict        | bytedict        | bytedict
          | ON
 5308 | 158961 | $lineitem |  15 | text255         | text255         | text255
          | ON
(16 rows)
```

STL_BCAST

記錄在執行播送資料的查詢步驟期間網路活動的相關資訊。網路流量是藉由資料列數、位元組數和封包數所擷取的，而這些資料列、位元組和封包是在特定配量上的特定步驟期間透過網路傳送的。步驟的持續時間是記錄的開始時間與結束時間之間的差異。

若要識別查詢中的播送步驟，請尋找 SVL_QUERY_SUMMARY 檢視中的 bcast 標籤，或執行 EXPLAIN 命令，然後尋找包含 bcast 的步驟屬性。

所有使用者都可看見 STL_BCAST。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_BCAST 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

欄名稱	資料類型	描述
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
packets	integer	透過網路傳送的封包總數。

範例查詢

下列範例傳回查詢的播送資訊，其中具有一個或多個封包，而且查詢的開始與結束之間的差異為一秒或以上。

```
select query, slice, step, rows, bytes, packets, datediff(seconds, starttime, endtime)
from stl_bcast
where packets>0 and datediff(seconds, starttime, endtime)>0;
```

```
 query | slice | step | rows | bytes | packets | date_diff
-------+-------+------+------+-------+---------+-----------
   453 |     2 |    5 |    1 |   264 |       1 |         1
   798 |     2 |    5 |    1 |   264 |       1 |         1
  1408 |     2 |    5 |    1 |   264 |       1 |         1
  2993 |     0 |    5 |    1 |   264 |       1 |         1
  5045 |     3 |    5 |    1 |   264 |       1 |         1
  8073 |     3 |    5 |    1 |   264 |       1 |         1
  8163 |     3 |    5 |    1 |   264 |       1 |         1
  9212 |     1 |    5 |    1 |   264 |       1 |         1
  9873 |     1 |    5 |    1 |   264 |       1 |         1
(9 rows)
```

STL_COMMIT_STATS

提供與遞交效能相關的指標，包括各種遞交階段的計時，以及遞交的區塊數。查詢 STL_COMMIT_STATS，以判斷交易的哪個部分花費在遞交上，以及發生多少佇列。

只有超級使用者才能看到 STL_COMMIT_STATS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_TRANSACTION_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
xid	bigint	交易 ID 遞交中。
node	integer	節點號碼。-1 是領導者節點。
startqueue	timestamp	開始遞交佇列。
startwork	timestamp	開始遞交。
endflush	timestamp	結束髒區塊排清階段。
endstage	timestamp	結束中繼資料臨時階段。
endlocal	timestamp	結束本機遞交階段。
startglobal	timestamp	開始全域階段。
endtime	timestamp	結束遞交。
queuelen	bigint	遞交佇列中在此交易之前的交易數。
permblocks	bigint	進行此遞交時現有永久區塊的數目。
newblocks	bigint	進行此遞交時新的永久區塊的數目。
dirtyblocks	bigint	必須寫入為此遞交一部分的區塊數目。
headers	bigint	必須寫入為此遞交一部分的區塊標頭數目。
numxids	integer	作用中 DML 交易的數目。
oldestxid	bigint	最舊作用中 DML 交易的 XID。
extwritelatency	bigint	此資訊僅供內部使用。

欄名稱	資料類型	描述
metadataw ritten	int	此資訊僅供內部使用。
tombstone dblocks	bigint	此資訊僅供內部使用。
tossedblo cks	bigint	此資訊僅供內部使用。
batched_by	bigint	此資訊僅供內部使用。

範例查詢

```
select node, datediff(ms,startqueue,startwork) as queue_time,
datediff(ms, startwork, endtime) as commit_time, queuelen
from stl_commit_stats
where xid = 2574
order by node;

node | queue_time   | commit_time | queuelen
-----+--------------+-------------+---------
  -1 |            0 |         617 |        0
   0 | 444950725641 |         616 |        0
   1 | 444950725636 |         616 |        0
```

STL_CONNECTION_LOG

記錄身分驗證嘗試以及連線和中斷連線。

只有超級使用者才能看到 STL_CONNECTION_LOG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_CONNECTION_LOG 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
事件	character(50)	連線或身分驗證事件。
recordtime	timestamp	發生事件的時間。
remotehost	character(45)	遠端主機的名稱或 IP 地址。
remoteport	character(32)	遠端主機的連接埠號碼。
pid	integer	與陳述式相關聯的處理程序 ID。
dbname	character(50)	資料庫名稱。
使用者名稱	character(50)	使用者名稱。
authmethod	character(32)	身分驗證方法。
持續時間	integer	連線的持續時間，以微秒為單位。
sslversion	character(50)	Secure Sockets Layer (SSL) 版本。
sslcipher	character(128)	SSL 密碼。
mtu	integer	最大傳輸單位 (MTU)。
sslcompression	character(64)	SSL 壓縮類型。
sslexpansion	character(64)	SSL 擴展類型。
iamauthguid	character(36)	請求的 IAM 身份驗證 CloudTrail ID。
application_name	character(250)	工作階段之應用程式的初始或已更新名稱。
os_version	character(64)	連線到 Amazon Redshift 叢集之用戶端機器上的作業系統版本。

欄名稱	資料類型	描述
driver_version	character(64)	從第三方 SQL 用戶端工具連線到 Amazon Redshift 叢集的 ODBC 或 JDBC 驅動器版本。
plugin_name	character(32)	連接至您的 Amazon Redshift 叢集時使用的外掛程式名稱。
protocol_version	integer	Amazon Redshift 驅動程式在建立與伺服器的連線時使用的內部通訊協定版本。通訊協定版本會在驅動程式與伺服器之間進行交涉。該版本描述了可用的功能。有效值包含： • 0 (BASE_SERVER_PROTOCOL_VERSION) • 1 (EXTENDED_RESULT_METADATA_SERVER_PROTOCOL_VERSION) - 為了儲存每個查詢的往返，伺服器會傳送額外的結果集中繼資料資訊。 • 2 (BINARY_PROTECOL_VERSION) - 根據結果集的資料類型，伺服器會以二進位格式傳送資料。 • 3 (EXTENDED2_RESULT_METADATA_SERVER_PROTOCOL_VERSION) - 伺服器傳送欄的區分大小寫 (定序) 資訊。
sessionid	character(36)	目前工作階段的全域唯一識別碼。工作階段 ID 在節點故障重新啟動後仍然存在。
compression	character(16)	用於連線的壓縮演算法。

範例查詢

若要檢視已開啟之連線的詳細資訊，請執行下列查詢。

```
select recordtime, username, dbname, remotehost, remoteport
from stl_connection_log
where event = 'initiating session'
and pid not in
(select pid from stl_connection_log
where event = 'disconnecting session')
```

```
order by 1 desc;

recordtime          | username    | dbname     | remotehost    | remoteport

--------------------+-------------+------------+---------------+------------
2014-11-06 20:30:06 | rdsdb       | dev        | [local]       |

2014-11-06 20:29:37 | test001     | test       | 10.49.42.138  | 11111

2014-11-05 20:30:29 | rdsdb       | dev        | 10.49.42.138  | 33333

2014-11-05 20:28:35 | rdsdb       | dev        | [local]       |
(4 rows)
```

下列範例反映失敗的身分驗證嘗試，以及成功的連線和中斷連線。

```
select event, recordtime, remotehost, username
from stl_connection_log order by recordtime;

            event      |         recordtime         |  remotehost   | username

-----------------------+----------------------------+---------------+---------
authentication failure | 2012-10-25 14:41:56.96391 | 10.49.42.138 | john

authenticated          | 2012-10-25 14:42:10.87613 | 10.49.42.138 | john

initiating session     | 2012-10-25 14:42:10.87638 | 10.49.42.138 | john

disconnecting session  | 2012-10-25 14:42:19.95992 | 10.49.42.138 | john

(4 rows)
```

下列範例顯示 ODBC 驅動程式的版本、用戶端機器上的作業系統，以及用來連線到 Amazon Redshift 叢集的外掛程式。在此範例中，使用的外掛程式用於使用登入名稱和密碼進行標準 ODBC 驅動程式驗證。

```
select driver_version, os_version, plugin_name from stl_connection_log;

driver_version                          |  os_version                        |
 plugin_name
-----------------------------------------+------------------------------------
+--------------------
```

```
Amazon Redshift ODBC Driver 1.4.15.0001 | Darwin 18.7.0 x86_64               | none
Amazon Redshift ODBC Driver 1.4.15.0001 | Linux 4.15.0-101-generic x86_64   | none
```

下列範例顯示用戶端電腦上的作業系統版本、驅動程式版本和通訊協定版本。

```
select os_version, driver_version, protocol_version from stl_connection_log;

os_version                       |  driver_version              | protocol_version
---------------------------------+------------------------------+-----------------
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
```

STL_DDLTEXT

擷取已在系統上執行的下列 DDL 陳述式。

這些 DDL 陳述式包括下列查詢和物件：

- CREATE SCHEMA、TABLE、VIEW
- DROP SCHEMA、TABLE、VIEW
- ALTER SCHEMA、TABLE

另請參閱 STL_QUERYTEXT、STL_UTILITYTEXT 和 SVL_STATEMENTTEXT。這些檢視提供在系統上執行之 SQL 命令的時間軸；此歷史記錄有助於進行故障診斷，以及建立所有系統活動的稽核記錄。

使用 STARTTIME 和 ENDTIME 資料欄，來了解已在特定時段記錄哪些陳述式。SQL 文字的長區塊會分成數行，一行 200 個字元；SEQUENCE 欄會識別屬於單一陳述式的文字片段。

所有使用者都可看見 STL_DDLTEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。

欄名稱	資料類型	描述
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位為空白。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回包含先前執行之 DDL 陳述式的記錄。

```
select xid, starttime, sequence, substring(text,1,40) as text
from stl_ddltext order by xid desc, sequence;
```

以下是顯示四個 CREATE TABLE 陳述式的範例輸出。DDL 陳述式會出現在 `text` 欄中，為了方便閱讀而被截斷。

```
 xid  |         starttime          | sequence |                  text
------+----------------------------+----------
+----------------------------------------
```

```
 1806 | 2013-10-23 00:11:14.709851 |        0 | CREATE TABLE supplier ( s_suppkey int4
 N
 1806 | 2013-10-23 00:11:14.709851 |        1 |  s_comment varchar(101) NOT NULL )
 1805 | 2013-10-23 00:11:14.496153 |        0 | CREATE TABLE region ( r_regionkey int4
 N
 1804 | 2013-10-23 00:11:14.285986 |        0 | CREATE TABLE partsupp ( ps_partkey int8
 1803 | 2013-10-23 00:11:14.056901 |        0 | CREATE TABLE part ( p_partkey int8 NOT
 N
 1803 | 2013-10-23 00:11:14.056901 |        1 | ner char(10) NOT NULL , p_retailprice
 nu
(6 rows)
```

重建儲存的 SQL

下列 SQL 會列出儲存在 STL_DDLTEXT 之 text 欄中的列。列依 xid 與 sequence 排序。如果原始 SQL 的多個列長度超過 200 個字元，則 STL_DDLTEXT 可以依 sequence 包含多個列。

```
SELECT xid, sequence, LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text)
 END, '') WITHIN GROUP (ORDER BY sequence) as query_statement
FROM stl_ddltext GROUP BY xid, sequence ORDER BY xid, sequence;
```

```
xid     |  sequence | query_statement
--------+-----------+-----------------
7886671    0          create external schema schema_spectrum_uddh\nfrom data catalog
\ndatabase 'spectrum_db_uddh'\niam_role ''\ncreate external database if not exists;
7886752    0          CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league\n(\n
  league_rank smallint,\n  prev_rank   smallint,\n  club_name   varchar(15),\n
 league_name varchar(20),\n  league_off  decimal(6,2),\n  le
7886752    1          ague_def  decimal(6,2),\n  league_spi  decimal(6,2),\n
 league_nspi smallint\n)\nROW FORMAT DELIMITED \n    FIELDS TERMINATED BY ',' \n
 LINES TERMINATED BY '\\n\\l'\nstored as textfile\nLOCATION 's
7886752    2          3://mybucket-spectrum-uddh/'\ntable properties
 ('skip.header.line.count'='1');
...
```

若要重建儲存在 STL_DDLTEXT 中 text 欄的 SQL，請執行下列 SQL 陳述式。它會將 text 欄中一或多個區段的 DDL 陳述式放在一起。請先在 SQL 用戶端中以新的一行取代任意 (\n) 特殊字元，再執行重建的 SQL。下列 SELECT 陳述式的結果會依序將三個列放在一起，以便在 query_statement 欄位中重建 SQL。

```
SELECT LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END) WITHIN
 GROUP (ORDER BY sequence) as query_statement
FROM stl_ddltext GROUP BY xid, endtime order by xid, endtime;
```

```
query_statement
--------------
create external schema schema_spectrum_uddh\nfrom data catalog\ndatabase
 'spectrum_db_uddh'\niam_role ''\ncreate external database if not exists;
CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league\n(\n  league_rank smallint,
\n  prev_rank   smallint,\n  club_name   varchar(15),\n  league_name varchar(20),\n
 league_off  decimal(6,2),\n  league_def  decimal(6,2),\n  league_spi  decimal(6,2),
\n  league_nspi smallint\n)\nROW FORMAT DELIMITED \n    FIELDS TERMINATED BY ',' \n
  LINES TERMINATED BY '\\n\\l'\nstored as textfile\nLOCATION 's3://mybucket-spectrum-
uddh/'\ntable properties ('skip.header.line.count'='1');
```

STL_DELETE

分析查詢的刪除執行步驟。

所有使用者都可看見 STL_DELETE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_DELETE 只包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。

欄名稱	資料類型	描述
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。

範例查詢

為了要在 STL_DELETE 中建立一個資料列，下列範例會將一個資料列插入至 EVENT 資料表，然後刪除它。

首先，將一個資料列插入至 EVENT 資料表，然後驗證是否已插入它。

```
insert into event(eventid,venueid,catid,dateid,eventname)
values ((select max(eventid)+1 from event),95,9,1857,'Lollapalooza');
```

```
select * from event
where eventname='Lollapalooza'
order by eventid;
```

```
 eventid | venueid | catid | dateid |  eventname   |      starttime
---------+---------+-------+--------+--------------+---------------------
    4274 |     102 |     9 |   1965 | Lollapalooza | 2008-05-01 19:00:00
    4684 |     114 |     9 |   2105 | Lollapalooza | 2008-10-06 14:00:00
    5673 |     128 |     9 |   1973 | Lollapalooza | 2008-05-01 15:00:00
```

```
    5740 |      51 |     9 |   1933 | Lollapalooza | 2008-04-17 15:00:00
    5856 |     119 |     9 |   1831 | Lollapalooza | 2008-01-05 14:00:00
    6040 |     126 |     9 |   2145 | Lollapalooza | 2008-11-15 15:00:00
    7972 |      92 |     9 |   2026 | Lollapalooza | 2008-07-19 19:30:00
    8046 |      65 |     9 |   1840 | Lollapalooza | 2008-01-14 15:00:00
    8518 |      48 |     9 |   1904 | Lollapalooza | 2008-03-19 15:00:00
    8799 |      95 |     9 |   1857 | Lollapalooza |
(10 rows)
```

現在，刪除您已新增至 EVENT 資料表的資料列，然後驗證是否已刪除它。

```
delete from event
where eventname='Lollapalooza' and eventid=(select max(eventid) from event);
```

```
select * from event
where eventname='Lollapalooza'
order by eventid;
```

```
 eventid | venueid | catid | dateid |  eventname   |      starttime
---------+---------+-------+--------+--------------+---------------------
    4274 |     102 |     9 |   1965 | Lollapalooza | 2008-05-01 19:00:00
    4684 |     114 |     9 |   2105 | Lollapalooza | 2008-10-06 14:00:00
    5673 |     128 |     9 |   1973 | Lollapalooza | 2008-05-01 15:00:00
    5740 |      51 |     9 |   1933 | Lollapalooza | 2008-04-17 15:00:00
    5856 |     119 |     9 |   1831 | Lollapalooza | 2008-01-05 14:00:00
    6040 |     126 |     9 |   2145 | Lollapalooza | 2008-11-15 15:00:00
    7972 |      92 |     9 |   2026 | Lollapalooza | 2008-07-19 19:30:00
    8046 |      65 |     9 |   1840 | Lollapalooza | 2008-01-14 15:00:00
    8518 |      48 |     9 |   1904 | Lollapalooza | 2008-03-19 15:00:00
(9 rows)
```

然後，查詢 stl_delete 以查看刪除的執行步驟。在此範例中，查詢已傳回超過 300 個的資料列，因此以下輸出會縮短以便於顯示。

```
select query, slice, segment, step, tasknum, rows, tbl from stl_delete order by query;
```

```
 query | slice | segment | step | tasknum | rows |  tbl
-------+-------+---------+------+---------+------+--------
     7 |     0 |       0 |    1 |       0 |    0 | 100000
```

```
 7 |     1 |       0 |    1 |       0 |    0 | 100000
 8 |     0 |       0 |    1 |       2 |    0 | 100001
 8 |     1 |       0 |    1 |       2 |    0 | 100001
 9 |     0 |       0 |    1 |       4 |    0 | 100002
 9 |     1 |       0 |    1 |       4 |    0 | 100002
10 |     0 |       0 |    1 |       6 |    0 | 100003
10 |     1 |       0 |    1 |       6 |    0 | 100003
11 |     0 |       0 |    1 |       8 |    0 | 100253
11 |     1 |       0 |    1 |       8 |    0 | 100253
12 |     0 |       0 |    1 |       0 |    0 | 100255
12 |     1 |       0 |    1 |       0 |    0 | 100255
13 |     0 |       0 |    1 |       2 |    0 | 100257
13 |     1 |       0 |    1 |       2 |    0 | 100257
14 |     0 |       0 |    1 |       4 |    0 | 100259
14 |     1 |       0 |    1 |       4 |    0 | 100259
...
```

STL_DISK_FULL_DIAG

磁碟已滿時記錄有關錯誤的日誌資訊。

只有超級使用者才能看到 STL_DISK_FULL_DIAG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述		
currenttime	bigint	自 2000 年 1 月 1 日起，產生錯誤的日期和時間 (以微秒為單位)。		
node_num	bigint	節點的識別碼。		
query_id	bigint	造成錯誤之查詢的識別碼。		
temp_blocks	bigint	查詢建立的臨時區塊數。		

範例查詢

下列範例會在發生磁碟已滿錯誤時，傳回所儲存資料的詳細資訊。

```
select * from stl_disk_full_diag
```

下列範例會將 currenttime 轉換為時間戳記。

```
select '2000-01-01'::timestamp + (currenttime/1000000.0)* interval '1 second' as
 currenttime,node_num,query_id,temp_blocks from pg_catalog.stl_disk_full_diag;
```

```
        currenttime         | node_num | query_id | temp_blocks
----------------------------+----------+----------+-------------
 2019-05-18 19:19:18.609338 |        0 |   569399 |       70982
 2019-05-18 19:37:44.755548 |        0 |   569580 |       70982
 2019-05-20 13:37:20.566916 |        0 |   597424 |       70869
```

STL_DIST

記錄在執行遞送資料的查詢步驟期間網路活動的相關資訊。網路流量是藉由資料列數、位元組數和封包數所擷取的，而這些資料列、位元組和封包是在特定配量上的特定步驟期間透過網路傳送的。步驟的持續時間是記錄的開始時間與結束時間之間的差異。

若要識別查詢中的配送步驟，請尋找 QUERY_SUMMARY 檢視中的 dist 標籤，或執行 EXPLAIN 命令，然後尋找包含 dist 的步驟屬性。

所有使用者都可看見 STL_DIST。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_DIST 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
packets	integer	透過網路傳送的封包總數。

範例查詢

下列範例傳回查詢的配送資訊，其中具有一個或多個封包，且持續時間大於零。

```
select query, slice, step, rows, bytes, packets,
datediff(seconds, starttime, endtime) as duration
from stl_dist
where packets>0 and datediff(seconds, starttime, endtime)>0
order by query
limit 10;
```

```
 query  | slice | step |  rows  |  bytes   | packets | duration
--------+-------+------+--------+----------+---------+----------
    567 |     1 |    4 |  49990 | 6249564 |     707 |         1
    630 |     0 |    5 |   8798 |  408404 |      46 |         2
    645 |     1 |    4 |   8798 |  408404 |      46 |         1
    651 |     1 |    5 | 192497 | 9226320 |    1039 |         6
    669 |     1 |    4 | 192497 | 9226320 |    1039 |         4
    675 |     1 |    5 |   3766 |  194656 |      22 |         1
    696 |     0 |    4 |   3766 |  194656 |      22 |         1
    705 |     0 |    4 |    930 |   44400 |       5 |         1
 111525 |     0 |    3 |     68 |   17408 |       2 |         1
(9 rows)
```

STL_ERROR

記錄 Amazon Redshift 資料庫引擎所產生的內部處理錯誤。STL_ERROR 不會記錄 SQL 錯誤或訊息。STL_ERROR 中的資訊有助於對特定錯誤進行故障診斷。 AWS 支援工程師可能會要求您提供此資訊，做為疑難排解程序的一部分。

所有使用者都可看見 STL_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需使用 Copy 命令載入資料時可能產生之錯誤代碼的清單，請參閱載入錯誤參考。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
process	character(12)	擲出例外狀況的處理程序。
recordtime	timestamp	發生錯誤的時間。
pid	integer	處理程序 ID。STL_QUERY 資料表包含處理程序 ID，以及已完成查詢的唯一查詢 ID。

欄名稱	資料類型	描述
errcode	integer	對應於錯誤類別的錯誤代碼。
file	character(90)	發生錯誤之來源檔案的名稱。
linenum	integer	來源檔案中發生錯誤的行號。
context	character(100)	錯誤的原因。
error	character(512)	錯誤訊息。

範例查詢

下列範例從 STL_ERROR 擷取錯誤資訊。

```
select process, errcode, linenum as line,
trim(error) as err
from stl_error;

   process    | errcode | line |                                 err
--------------+---------+------
+-----------------------------------------------------------------
 padbmaster   |    8001 |  194 | Path prefix: s3://redshift-downloads/testnulls/
venue.txt*
 padbmaster   |    8001 |  529 | Listing bucket=redshift-downloads prefix=tests/
category-csv-quotes
 padbmaster   |       2 |  190 | database "template0" is not currently accepting
 connections
 padbmaster   |      32 | 1956 | pq_flush: could not send data to client: Broken pipe
(4 rows)
```

STL_EXPLAIN

顯示已提交供執行之用的查詢的 EXPLAIN 計劃。

所有使用者都可看見 STL_EXPLAIN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_EXPLAIN 僅包含在主要叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別碼，其中節點會在查詢執行時映射至一個或多個步驟。
parentid	integer	父節點的計劃節點識別碼。父節點具有一些子節點。例如，合併聯結是已聯結資料表上掃描的父項。
plannode	character(400)	來自 EXPLAIN 輸出的節點文字。在 EXPLAIN 輸出中，提及在運算節點上執行的計劃節點是以 **XN** 為字首。
info	character(400)	計劃節點的限定詞和篩選條件資訊。例如，此資料欄中包括聯結條件和 WHERE 子句限制。

範例查詢

考慮彙總聯結查詢的下列 EXPLAIN 輸出：

```
explain select avg(datediff(day, listtime, saletime)) as avgwait
from sales, listing where sales.listid = listing.listid;
                                QUERY PLAN

------------------------------------------------------------------------------
 XN Aggregate  (cost=6350.30..6350.31 rows=1 width=16)
  ->  XN Hash Join DS_DIST_NONE  (cost=47.08..6340.89 rows=3766 width=16)
        Hash Cond: ("outer".listid = "inner".listid)
```

```
        -> XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=12)
        -> XN Hash  (cost=37.66..37.66 rows=3766 width=12)
              -> XN Seq Scan on sales  (cost=0.00..37.66 rows=3766 width=12)
(6 rows)
```

如果執行此查詢且其查詢 ID 為 10，則您可以使用 STL_EXPLAIN 資料表，來查看 EXPLAIN 命令所傳回相同類型的資訊：

```
select query,nodeid,parentid,substring(plannode from 1 for 30),
substring(info from 1 for 20) from stl_explain
where query=10 order by 1,2;

query| nodeid |parentid|           substring            |    substring
-----+--------+--------+--------------------------------+-------------------
10   |      1 |      0 |XN Aggregate  (cost=6717.61..6  |
10   |      2 |      1 |  -> XN Merge Join DS_DIST_NO   | Merge Cond:("outer"
10   |      3 |      2 |       -> XN Seq Scan on lis    |
10   |      4 |      2 |       -> XN Seq Scan on sal    |
(4 rows)
```

請考處下列查詢：

```
select event.eventid, sum(pricepaid)
from event, sales
where event.eventid=sales.eventid
group by event.eventid order by 2 desc;

eventid |   sum
--------+----------
    289 | 51846.00
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
...
```

如果此查詢的 ID 為 15，則下列系統檢視查詢會傳回已完成的計劃節點。在此情況下，會保留節點的順序，以顯示執行的實際順序：

```
select query,nodeid,parentid,substring(plannode from 1 for 56)
from stl_explain where query=15 order by 1, 2 desc;
```

```
query|nodeid|parentid|                              substring
-----+------+--------+-------------------------------------------------------------
15   |    8 |      7 |                                    -> XN Seq Scan on eve
15   |    7 |      5 |                               -> XN Hash(cost=87.98..87.9
15   |    6 |      5 |                               -> XN Seq Scan on sales(cos
15   |    5 |      4 |                         -> XN Hash Join DS_DIST_OUTER(cos
15   |    4 |      3 |                  -> XN HashAggregate(cost=862286577.07..
15   |    3 |      2 |           -> XN Sort(cost=1000862287175.47..10008622871
15   |    2 |      1 |  -> XN Network(cost=1000862287175.47..1000862287197.
15   |    1 |      0 |XN Merge(cost=1000862287175.47..1000862287197.46 rows=87
(8 rows)
```

下列查詢會擷取包含視窗函數之任何查詢計劃的查詢 ID：

```
select query, trim(plannode) from stl_explain
where plannode like '%Window%';

query|                                       btrim
-----+-------------------------------------------------------------------------
26   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
27   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
(2 rows)
```

STL_FILE_SCAN

傳回 Amazon Redshift 在使用 COPY 命令載入資料時讀取的檔案。

查詢此檢視有助於對資料載入錯誤進行故障診斷。STL_FILE_SCAN 特別有助於指出平行資料載入中的問題，因為平行資料載入通常會利用單一 COPY 命令載入多個檔案。

所有使用者都可看見 STL_FILE_SCAN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_FILE_SCAN 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_LOAD_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
name	character(90)	已載入之檔案的完整路徑和名稱。
lines	bigint	從檔案讀取的行數。
位元組	bigint	從檔案讀取的位元組數。
loadtime	bigint	載入檔案所花費的時間 (以微秒為單位)。
curtime	Timestamp	代表 Amazon Redshift 開始處理檔案之時間的時間戳記。
is_partial	integer	如果值為 true (1)，則表示輸入檔在 COPY 操作期間被分割為多個範圍。如果此值為 false (0)，則輸入檔案不會分割。
start_offset	bigint	如果輸入檔案在 COPY 操作期間被分割，則該值指示分割的偏移值 (以位元組為單位)。如果檔案未分割，則此值為 0。

範例查詢

下列查詢擷取 Amazon Redshift 需要超過 1,000,000 微秒來讀取之任何檔案的名稱和載入時間。

```
select trim(name)as name, loadtime from stl_file_scan
where loadtime > 1000000;
```

此查詢傳回下列範例輸出。

```
          name             | loadtime
---------------------------+----------
 listings_pipe.txt         |  9458354
 allusers_pipe.txt         |  2963761
 allevents_pipe.txt        |  1409135
 tickit/listings_pipe.txt  |  7071087
```

```
 tickit/allevents_pipe.txt |  1237364
 tickit/allusers_pipe.txt  |  2535138
 listings_pipe.txt         |  6706370
 allusers_pipe.txt         |  3579461
 allevents_pipe.txt        |  1313195
 tickit/allusers_pipe.txt  |  3236060
 tickit/listings_pipe.txt  |  4980108
(11 rows)
```

STL_HASH

分析查詢的雜湊執行步驟。

所有使用者都可看見 STL_HASH。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_HASH 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

欄名稱	資料類型	描述
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
slots	integer	雜湊儲存貯體的總數。
occupied	integer	包含記錄之插槽的總數。
maxlength	integer	最大插槽的大小。
tbl	integer	表格 ID。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體的位元組總數。
num_parts	integer	在雜湊步驟期間雜湊資料表已劃分的分割區總數。
est_rows	bigint	要雜湊之資料列的預估數目。
num_blocks_permitted	integer	此資訊僅供內部使用。
resizes	integer	此資訊僅供內部使用。
checksum	bigint	此資訊僅供內部使用。
runtime_filter_size	integer	執行時間篩選條件的大小，以位元組為單位。

欄名稱	資料類型	描述
max_runtime_filter_size	integer	執行時間篩選條件的大小上限，以位元組為單位。

範例查詢

下列範例會傳回雜湊中針對查詢 720 所使用之分割區數目的相關資訊，並指出已沒有步驟在磁碟上執行。

```
select slice, rows, bytes, occupied, workmem, num_parts, est_rows,
 num_blocks_permitted, is_diskbased
from stl_hash
where query=720 and segment=5
order by slice;
```

```
 slice | rows | bytes  | occupied | workmem  | num_parts | est_rows |
 num_blocks_permitted | is_diskbased
-------+------+--------+----------+----------+-----------+----------
+----------------------+--------------
     0 |  145 | 585800 |        1 | 88866816 |        16 |        1 |
 52                   f
     1 |    0 |      0 |        0 |        0 |        16 |        1 |
 52                   f
(2 rows)
```

STL_HASHJOIN

分析查詢的雜湊聯結執行步驟。

所有使用者都可看見 STL_HASHJOIN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_HASHJOIN 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。
num_parts	integer	在雜湊步驟期間雜湊資料表已劃分的分割區總數。
join_type	integer	步驟的聯結類型： • 0. 查詢已使用內部聯結。 • 1. 查詢已使用左外部聯結。 • 2. 查詢已使用完整外部聯結。 • 3. 查詢已使用右外部聯結。 • 4. 查詢已使用 UNION 操作。 • 5. 查詢已使用 IN 條件。 • 6. 此資訊僅供內部使用。

欄名稱	資料類型	描述
		• 7. 此資訊僅供內部使用。 • 8. 此資訊僅供內部使用。 • 9. 此資訊僅供內部使用。 • 10. 此資訊僅供內部使用。 • 11. 此資訊僅供內部使用。 • 12. 此資訊僅供內部使用。
hash_looped	character(1)	此資訊僅供內部使用。
switched_parts	character(1)	指出建置 (或外部) 和探測 (或內部) 端是否已切換。
used_prefetching	character(1)	此資訊僅供內部使用。
hash_segment	integer	對應雜湊步驟的區段。
hash_step	integer	對應雜湊步驟的步驟號碼。
checksum	bigint	此資訊僅供內部使用。
分佈	integer	此資訊僅供內部使用。

範例查詢

下列查詢會傳回雜湊聯結中針對查詢 720 所使用的分割區數目。

```
select query, slice, tbl, num_parts
from stl_hashjoin
where query=720 limit 10;
```

```
 query | slice | tbl | num_parts
-------+-------+-----+-----------
   720 |     0 | 243 |         1
   720 |     1 | 243 |         1
```

```
(2 rows)
```

STL_INSERT

分析查詢的插入執行步驟。

所有使用者都可看見 STL_INSERT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_INSERT 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。

欄名稱	資料類型	描述
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。
inserted_mega_value	character(1)	此資訊僅供內部使用。此資訊顯示指定的插入步驟是否已插入較大的值。大值將儲存在多個區塊中。區塊大小預設為 1 MB，在預設設定中，大值大於 1 MB。

範例查詢

下列範例會傳回最新查詢的插入執行步驟。

```
select slice, segment, step, tasknum, rows, tbl
from stl_insert
where query=pg_last_query_id();
```

```
 slice | segment | step | tasknum | rows  |  tbl
-------+---------+------+---------+-------+--------
     0 |       2 |    2 |      15 | 24958 | 100548
     1 |       2 |    2 |      15 | 25032 | 100548
(2 rows)
```

STL_LIMIT

分析在 SELECT 查詢中使用 LIMIT 子句時發生的執行步驟。

所有使用者都可看見 STL_LIMIT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_LIMIT 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

為了要在 STL_LIMIT 中產生資料列，此範例會使用 LIMIT 子句，針對 VENUE 資料表執行下列查詢。

```
select * from venue
order by 1
limit 10;
```

```
 venueid |         venuename          |    venuecity    | venuestate | venueseats
---------+----------------------------+-----------------+------------+------------
       1 | Toyota Park                | Bridgeview      | IL         |          0
       2 | Columbus Crew Stadium      | Columbus        | OH         |          0
```

```
       3 | RFK Stadium                | Washington      | DC         |          0
       4 | CommunityAmerica Ballpark  | Kansas City     | KS         |          0
       5 | Gillette Stadium           | Foxborough      | MA         |      68756
       6 | New York Giants Stadium    | East Rutherford | NJ         |      80242
       7 | BMO Field                  | Toronto         | ON         |          0
       8 | The Home Depot Center      | Carson          | CA         |          0
       9 | Dick's Sporting Goods Park | Commerce City   | CO         |          0
      10 | Pizza Hut Park             | Frisco          | TX         |          0
(10 rows)
```

接著，執行下列查詢，以尋找您上次針對 VENUE 資料表所執行之查詢的查詢 ID。

```
select max(query)
from stl_query;
```

```
  max
--------
 127128
(1 row)
```

您可以選擇性地執行下列查詢，以驗證查詢 ID 對應至您先前執行的 LIMIT 查詢。

```
select query, trim(querytxt)
from stl_query
where query=127128;
```

```
 query  |                 btrim
--------+----------------------------------------
 127128 | select * from venue order by 1 limit 10;
(1 row)
```

最後，執行下列查詢，從 STL_LIMIT 資料表傳回 LIMIT 查詢的相關資訊。

```
select slice, segment, step, starttime, endtime, tasknum
from stl_limit
where query=127128
order by starttime, endtime;
```

```
  slice | segment | step |         starttime          |          endtime           |
 tasknum
```

```
 -------+---------+------+---------------------------+---------------------------
+---------
      1 |       1 |    3 | 2013-09-06 22:56:43.608114 | 2013-09-06 22:56:43.609383 |
   15
      0 |       1 |    3 | 2013-09-06 22:56:43.608708 | 2013-09-06 22:56:43.609521 |
   15
  10000 |       2 |    2 | 2013-09-06 22:56:43.612506 | 2013-09-06 22:56:43.612668 |
    0
(3 rows)
```

STL_LOAD_COMMITS

傳回資訊以追蹤資料載入或對其進行故障診斷。

此檢視會在每一個資料檔案載入至資料庫資料表時記錄其進度。

所有使用者都可看見 STL_LOAD_COMMITS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_LOAD_COMMITS 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_LOAD_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	針對此項目載入的配量。
name	character(256)	系統定義的值。
filename	character(256)	正在追蹤之檔案的名稱。
byte_offset	integer	此資訊僅供內部使用。

欄名稱	資料類型	描述
lines_scanned	integer	從載入檔案掃描的行數。此數目可能不符合實際載入的資料列數目。例如，根據 COPY 命令中的 MAXERROR 選項，載入可能掃描但容忍一些錯誤記錄。
錯誤	integer	此資訊僅供內部使用。
curtime	timestamp	上次更新此項目的時間。
status	integer	此資訊僅供內部使用。
file_format	character(16)	載入檔案的格式。可能的值如下： • Avro • JSON • ORC • Parquet • 文字
is_partial	integer	如果值為 true (1)，則表示輸入檔在 COPY 操作期間被分割為多個範圍。如果此值為 false (0)，則輸入檔案不會分割。
start_offset	bigint	如果輸入檔案在 COPY 操作期間被分割，則該值指示分割的偏移值 (以位元組為單位)。每個檔案分割都記錄為具有對應 start_offset 值的單獨記錄。如果檔案未分割，則此值為 0。
copy_job_id	bigint	複製任務識別碼。0 表示沒有任務識別碼。

範例查詢

下列範例傳回上次 COPY 操作的詳細資訊。

```
select query, trim(filename) as file, curtime as updated
from stl_load_commits
where query = pg_last_copy_id();

 query |               file                |          updated
-------+-----------------------------------+----------------------------
```

```
 28554 | s3://dw-tickit/category_pipe.txt | 2013-11-01 17:14:52.648486
(1 row)
```

下列查詢包含 TICKIT 資料庫中全新載入之資料表的項目：

```
select query, trim(filename), curtime
from stl_load_commits
where filename like '%tickit%' order by query;
```

```
 query |           btrim           |          curtime
-------+---------------------------+---------------------------
 22475 | tickit/allusers_pipe.txt  | 2013-02-08 20:58:23.274186
 22478 | tickit/venue_pipe.txt     | 2013-02-08 20:58:25.070604
 22480 | tickit/category_pipe.txt  | 2013-02-08 20:58:27.333472
 22482 | tickit/date2008_pipe.txt  | 2013-02-08 20:58:28.608305
 22485 | tickit/allevents_pipe.txt | 2013-02-08 20:58:29.99489
 22487 | tickit/listings_pipe.txt  | 2013-02-08 20:58:37.632939
 22593 | tickit/allusers_pipe.txt  | 2013-02-08 21:04:08.400491
 22596 | tickit/venue_pipe.txt     | 2013-02-08 21:04:10.056055
 22598 | tickit/category_pipe.txt  | 2013-02-08 21:04:11.465049
 22600 | tickit/date2008_pipe.txt  | 2013-02-08 21:04:12.461502
 22603 | tickit/allevents_pipe.txt | 2013-02-08 21:04:14.785124
 22605 | tickit/listings_pipe.txt  | 2013-02-08 21:04:20.170594

(12 rows)
```

事實上，記錄寫入至此系統檢視的日誌檔案，並不表示已成功遞交載入，做為其包含交易的一部分。若要驗證載入遞交，請查詢 STL_UTILITYTEXT 檢視，並尋找與 COPY 交易相對應的 COMMIT 記錄。例如，此查詢會針對 STL_UTILITYTEXT 根據子查詢聯結 STL_LOAD_COMMITS 和 STL_QUERY：

```
select l.query,rtrim(l.filename),q.xid
from stl_load_commits l, stl_query q
where l.query=q.query
and exists
(select xid from stl_utilitytext where xid=q.xid and rtrim("text")='COMMIT');

 query |           rtrim           |  xid
-------+---------------------------+-------
 22600 | tickit/date2008_pipe.txt  | 68311
 22480 | tickit/category_pipe.txt  | 68066
  7508 | allusers_pipe.txt         | 23365
  7552 | category_pipe.txt         | 23415
```

```
  7576 | allevents_pipe.txt        | 23429
  7516 | venue_pipe.txt            | 23390
  7604 | listings_pipe.txt         | 23445
 22596 | tickit/venue_pipe.txt     | 68309
 22605 | tickit/listings_pipe.txt  | 68316
 22593 | tickit/allusers_pipe.txt  | 68305
 22485 | tickit/allevents_pipe.txt | 68071
  7561 | allevents_pipe.txt        | 23429
  7541 | category_pipe.txt         | 23415
  7558 | date2008_pipe.txt         | 23428
 22478 | tickit/venue_pipe.txt     | 68065
   526 | date2008_pipe.txt         |  2572
  7466 | allusers_pipe.txt         | 23365
 22482 | tickit/date2008_pipe.txt  | 68067
 22598 | tickit/category_pipe.txt  | 68310
 22603 | tickit/allevents_pipe.txt | 68315
 22475 | tickit/allusers_pipe.txt  | 68061
   547 | date2008_pipe.txt         |  2572
 22487 | tickit/listings_pipe.txt  | 68072
  7531 | venue_pipe.txt            | 23390
  7583 | listings_pipe.txt         | 23445
(25 rows)
```

以下範例反白顯示 is_partial 和 start_offset 欄值。

```
-- Single large file copy without scan range
SELECT count(*) FROM stl_load_commits WHERE query = pg_last_copy_id();
1

-- Single large uncompressed, delimited file copy with scan range
SELECT count(*) FROM stl_load_commits WHERE query = pg_last_copy_id();
16

-- Scan range offset logging in the file at 64MB boundary.
SELECT start_offset FROM stl_load_commits
WHERE query = pg_last_copy_id() ORDER BY start_offset;
0
67108864
134217728
201326592
268435456
335544320
402653184
```

```
469762048
536870912
603979776
671088640
738197504
805306368
872415232
939524096
1006632960
```

STL_LOAD_ERRORS

顯示所有 Amazon Redshift 載入錯誤的記錄。

STL_LOAD_ERRORS 包含所有 Amazon Redshift 載入錯誤的歷史記錄。如需可能載入錯誤和說明的完整清單，請參閱載入錯誤參考。

在您查詢 STL_LOAD_ERRORS 以了解有關錯誤的一般資訊之後，查詢 STL_LOADERROR_DETAIL 以取得其他詳細資訊，例如發生剖析錯誤的確切資料列和欄。

所有使用者都可看見 STL_LOAD_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_LOAD_ERRORS 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_LOAD_ERROR_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
分割	integer	發生錯誤的配量。
tbl	integer	表格 ID。
starttime	timestamp	載入的開始時間，以 UTC 表示。

欄名稱	資料類型	描述
session	integer	執行載入之工作階段的工作階段 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
filename	character(256)	用於載入之輸入檔案的完整路徑。
line_number	bigint	載入檔案中發生錯誤的行號。對於來自 JSON 的 COPY，JSON 物件中發生錯誤之最後一行的行號。
colname	character(127)	發生錯誤的欄位。
type	character(10)	欄位的資料類型。
col_length	character(10)	資料欄長度 (如適用)。當資料類型具有限制長度時，會填入此欄位。例如，對於資料類型為 "character(3)" 的資料欄，此資料欄將包含值 "3"。
position	integer	欄位中錯誤的位置。
raw_line	character(1024)	包含錯誤的原始載入資料。載入資料中的多位元組字元會取代為句點。
raw_field_value	char(1024)	導致剖析錯誤之欄位 "colname" 的預先剖析值。
err_code	integer	錯誤代碼。
err_reason	character(100)	錯誤的說明。
is_partial	integer	如果值為 true (1)，則表示輸入檔在 COPY 操作期間被分割為多個範圍。如果此值為 false (0)，則輸入檔案不會分割。
start_offset	bigint	如果輸入檔案在 COPY 操作期間被分割，則該值指示分割的偏移值 (以位元組為單位)。如果檔案中的行號不明，則行號為 -1。如果檔案未分割，則此值為 0。
copy_job_id	bigint	複製任務識別碼。0 表示沒有任務識別碼。

範例查詢

下列查詢會將 STL_LOAD_ERRORS 聯結至 STL_LOADERROR_DETAIL，以檢視最近載入期間發生之錯誤的詳細資訊。

```
select d.query, substring(d.filename,14,20),
d.line_number as line,
substring(d.value,1,16) as value,
substring(le.err_reason,1,48) as err_reason
from stl_loaderror_detail d, stl_load_errors le
where d.query = le.query
and d.query = pg_last_copy_id();

 query |    substring      | line |  value   |             err_reason
-------+-------------------+------+----------+----------------------------
    558| allusers_pipe.txt |  251 | 251      | String contains invalid or
                                              unsupported UTF8 code
    558| allusers_pipe.txt |  251 | ZRU29FGR | String contains invalid or
                                              unsupported UTF8 code
    558| allusers_pipe.txt |  251 | Kaitlin  | String contains invalid or
                                              unsupported UTF8 code
    558| allusers_pipe.txt |  251 | Walter   | String contains invalid or
                                              unsupported UTF8 code
```

下列範例會使用 STL_LOAD_ERRORS 與 STV_TBL_PERM 搭配來建立新檢視，然後使用該檢視來判斷將資料載入至 EVENT 資料表時發生哪些錯誤：

```
create view loadview as
(select distinct tbl, trim(name) as table_name, query, starttime,
trim(filename) as input, line_number, colname, err_code,
trim(err_reason) as reason
from stl_load_errors sl, stv_tbl_perm sp
where sl.tbl = sp.id);
```

接著，下列查詢實際上會傳回上次載入 EVENT 資料表時發生的錯誤：

```
select table_name, query, line_number, colname, starttime,
trim(reason) as error
from loadview
where table_name ='event'
order by line_number limit 1;
```

此查詢會傳回 EVENT 資料表上次發生的載入錯誤。如果未發生任何載入錯誤，則查詢不會傳回任何資料列。在此範例中，查詢會傳回單一錯誤：

```
 table_name | query | line_number | colname | error | starttime
------+-----+----+----+-----------------------------------------------------------
+---------------------
event | 309 |  0 |  5 | Error in Timestamp value or format [%Y-%m-%d %H:%M:%S] |
 2014-04-22 15:12:44

(1 row)
```

如果 COPY 命令會自動分割大型、未壓縮、文字分隔的檔案資料以促進平行處理，則 line_number、is_partial 和 start_offset 欄會顯示分割的相關資訊。(如果原始檔案中的行號不可用，則行號可能是未知的。)

```
--scan ranges information
SELECT line_number, POSITION, btrim(raw_line), btrim(raw_field_value),
btrim(err_reason), is_partial, start_offset FROM stl_load_errors
WHERE query = pg_last_copy_id();

--result
-1,51,"1008771|13463413|463414|2|28.00|38520.72|0.06|0.07|NO|1998-08-30|1998-09-25|
1998-09-04|TAKE BACK RETURN|RAIL|ans cajole sly","NO","Char length exceeds DDL
 length",1,67108864
```

STL_LOADERROR_DETAIL

顯示當使用 COPY 命令來載入資料表時發生之資料剖析錯誤的日誌。若要保留磁碟空間，針對每個載入操作，每個節點配量最多可記錄 20 個錯誤。

將資料列中的欄位載入至資料表時，若 Amazon Redshift 無法剖析該欄位，即會發生剖析錯誤。例如，如果資料表資料欄預期 integer 資料類型，但資料檔案在該欄位中包含一串字母，則其會導致剖析錯誤。

在您查詢 STL_LOAD_ERRORS 以了解有關錯誤的一般資訊之後，查詢 STL_LOADERROR_DETAIL 以取得其他詳細資訊，例如發生剖析錯誤的確切資料列和資料欄。

STL_LOADERROR_DETAIL 檢視包含發生剖析錯誤的欄 (含) 之前的所有資料欄。使用 VALUE 欄位來查看實際上已在此資料欄中剖析的資料值，包括已正確剖析至錯誤的資料欄。

所有使用者都可看見此檢視。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_LOADERROR_DETAIL 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_LOAD_ERROR_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
分割	integer	發生錯誤的配量。
session	integer	執行載入之工作階段的工作階段 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
filename	character(256)	用於載入之輸入檔案的完整路徑。
line_number	bigint	載入檔案中發生錯誤的行號。
field	integer	發生錯誤的欄位。
colname	character(1024)	資料欄名稱。
value	character(1024)	欄位的已剖析資料值。(可能已截斷。) 載入資料中的多位元組字元會取代為句點。
is_null	integer	已剖析值是否為 null。
type	character(10)	欄位的資料類型。
col_length	character(10)	資料欄長度 (如適用)。當資料類型具有限制長度時，會填入此欄位。例如，對於資料類型為 "character(3)" 的資料欄，此資料欄將包含值 "3"。

範例查詢

下列查詢會將 STL_LOAD_ERRORS 聯結至 STL_LOADERROR_DETAIL，以檢視載入 EVENT 資料表 (資料表 ID 為 100133) 時發生之剖析錯誤的詳細資訊：

```
select d.query, d.line_number, d.value,
le.raw_line, le.err_reason
from stl_loaderror_detail d, stl_load_errors le
where
d.query = le.query
and tbl = 100133;
```

下列範例輸出會顯示成功載入的資料欄，包括發生錯誤的資料欄。在此範例中，於第三個資料欄中發生剖析錯誤之前已成功載入兩個資料欄，此錯誤是對於預期整數的欄位，卻不正確地剖析字元字串。因為欄位預期整數，所以它將字串 "aaa" (未初始化的資料) 剖析為 null，並產生剖析錯誤。輸出顯示原始值、已剖析值和錯誤原因：

```
query  | line_number | value | raw_line | err_reason
-------+-------------+-------+----------+----------------
4      |      3      |  1201 |  1201    | Invalid digit
4      |      3      |   126 |   126    | Invalid digit
4      |      3      |       |   aaa    | Invalid digit
(3 rows)
```

當查詢聯結 STL_LOAD_ERRORS 和 STL_LOADERROR_DETAIL 時，它會顯示資料列中每個資料欄的錯誤原因，這僅表示該資料列中發生錯誤。結果中的最後一列是發生剖析錯誤的實際資料欄。

STL_MERGE

分析查詢的合併執行步驟。當合併平行操作 (例如排序和聯結) 的結果進行後續處理時，即會進行這些步驟。

所有使用者都可看見 STL_MERGE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_MERGE 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。

範例查詢

下列範例會傳回 10 個合併執行結果。

```
select query, step, starttime, endtime, tasknum, rows
from stl_merge
limit 10;
```

```
 query | step |       starttime     |        endtime      | tasknum | rows
-------+------+---------------------+---------------------+---------+------
     9 |    0 | 2013-08-12 20:08:14 | 2013-08-12 20:08:14 |       0 |    0
    12 |    0 | 2013-08-12 20:09:10 | 2013-08-12 20:09:10 |       0 |    0
    15 |    0 | 2013-08-12 20:10:24 | 2013-08-12 20:10:24 |       0 |    0
    20 |    0 | 2013-08-12 20:11:27 | 2013-08-12 20:11:27 |       0 |    0
```

```
    26 |    0 | 2013-08-12 20:12:28 | 2013-08-12 20:12:28 |       0 |    0
    32 |    0 | 2013-08-12 20:14:33 | 2013-08-12 20:14:33 |       0 |    0
    38 |    0 | 2013-08-12 20:16:43 | 2013-08-12 20:16:43 |       0 |    0
    44 |    0 | 2013-08-12 20:17:05 | 2013-08-12 20:17:05 |       0 |    0
    50 |    0 | 2013-08-12 20:18:48 | 2013-08-12 20:18:48 |       0 |    0
    56 |    0 | 2013-08-12 20:20:48 | 2013-08-12 20:20:48 |       0 |    0
(10 rows)
```

STL_MERGEJOIN

分析查詢的合併聯結執行步驟。

所有使用者都可看見 STL_MERGEJOIN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_MERGEJOIN 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

欄名稱	資料類型	描述
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。這是已在合併聯結中使用之內部資料表的 ID。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例會傳回最新查詢的合併聯結結果。

```
select sum(s.qtysold), e.eventname
from event e, listing l, sales s
where e.eventid=l.eventid
and l.listid= s.listid
group by e.eventname;

select * from stl_mergejoin where query=pg_last_query_id();
```

```
 userid | query | slice | segment | step |         starttime   |          endtime    |
 tasknum | rows | tbl
--------+-------+-------+---------+------+---------------------+---------------------
+---------+------+-----
    100 | 27399 |     3 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |43428 | 240
    100 | 27399 |     0 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |43159 | 240
    100 | 27399 |     2 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |42778 | 240
    100 | 27399 |     1 |       4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
     19 |43091 | 240
```

STL_MV_STATE

對於具體化視觀表的每個狀態轉換，STL_MV_STATE 檢視都會包含一個列。

如需具體化視觀表的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

所有使用者都可看見 STL_MV_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_MV_STATE 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	bigint	建立事件的使用者 ID。
starttime	timestamp	事件的開始時間。
xid	bigint	事件的交易 ID。
event_desc	char(500)	提示狀態變更的事件。一些範例值包括如下： • 已變更資料欄類型 • 已捨棄資料欄 • 已重新命名資料欄 • 已變更結構描述名稱 • 小型資料表轉換 • TRUNCATE • Vacuum 請注意，此欄還有其他可能的值。
db_name	char(128)	包含具體化視觀表的資料庫。
base_table_schema	char(128)	基底資料表的結構描述。

欄名稱	資料類型	描述
base_table_name	char(128)	基底資料表的名稱。
mv_schema	char(128)	具體化視觀表的結構描述。
mv_name	char(128)	具體化視觀表的名稱。
state	character(32)	具體化視觀表的變更後狀態如下： • 重新運算 • 無法重新整理

下表顯示 event_desc 和 state 範例組合。

```
          event_desc     |     state
-------------------------+---------------
 TRUNCATE                | Recompute
 TRUNCATE                | Recompute
 Small table conversion  | Recompute
 Vacuum                  | Recompute
 Column was renamed      | Unrefreshable
 Column was dropped      | Unrefreshable
 Table was renamed       | Unrefreshable
 Column type was changed | Unrefreshable
 Schema name was changed | Unrefreshable
```

範例查詢

若要檢視具體化視觀表的狀態轉換日誌，請執行下列查詢。

```
select * from stl_mv_state;
```

此查詢傳回下列範例輸出：

```
 userid |         starttime          | xid  |            event_desc       | db_name |
 base_table_schema   |   base_table_name    |      mv_schema       | mv_name       |
  state
--------+----------------------------+------+-----------------------------+---------
+---------------------+---------------------+---------------------+--------------
+---------------
    138 | 2020-02-14 02:21:25.578885 | 5180 | TRUNCATE                    | dev     |
 public               | mv_base_table        | public               | mv_test       |
 Recompute
    138 | 2020-02-14 02:21:56.846774 | 5275 | Column was dropped          | dev     |
                      | mv_base_table        | public               | mv_test       |
 Unrefreshable
    100 | 2020-02-13 22:09:53.041228 | 1794 | Column was renamed          | dev     |
                      | mv_base_table        | public               | mv_test       |
 Unrefreshable
      1 | 2020-02-13 22:10:23.630914 | 1893 | ALTER TABLE ALTER SORTKEY   | dev     |
 public               | mv_base_table_sorted | public               | mv_test       |
 Recompute
      1 | 2020-02-17 22:57:22.497989 | 8455 | ALTER TABLE ALTER DISTSTYLE | dev     |
 public               | mv_base_table        | public               | mv_test       |
 Recompute
    173 | 2020-02-17 22:57:23.591434 | 8504 | Table was renamed           | dev     |
                      | mv_base_table        | public               | mv_test       |
 Unrefreshable
    173 | 2020-02-17 22:57:27.229423 | 8592 | Column type was changed     | dev     |
                      | mv_base_table        | public               | mv_test       |
 Unrefreshable
    197 | 2020-02-17 22:59:06.212569 | 9668 | TRUNCATE                    | dev     |
 schemaf796e415850f4f | mv_base_table        | schemaf796e415850f4f | mv_test       |
 Recompute
    138 | 2020-02-14 02:21:55.705655 | 5226 | Column was renamed          | dev     |
                      | mv_base_table        | public               | mv_test       |
 Unrefreshable
      1 | 2020-02-14 02:22:26.292434 | 5325 | ALTER TABLE ALTER SORTKEY   | dev     |
 public               | mv_base_table_sorted | public               | mv_test       |
 Recompute
```

STL_NESTLOOP

分析查詢的巢狀迴路聯結執行步驟。

所有使用者都可看見 STL_NESTLOOP。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_NESTLOOP 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。
checksum	bigint	此資訊僅供內部使用。

範例查詢

由於下列查詢會忽略聯結 CATEGORY 資料表，因此它會產生局部 Cartesian 產品，而我們不建議這樣做。在這裡顯示它，是為了說明巢狀迴路。

```
select count(event.eventname), event.eventname, category.catname, date.caldate
from event, category, date
where event.dateid = date.dateid
group by event.eventname, category.catname, date.caldate;
```

下列查詢會將來自前一個查詢的結果顯示在 STL_NESTLOOP 檢視中。

```
select query, slice, segment as seg, step,
datediff(msec, starttime, endtime) as duration, tasknum, rows, tbl
from stl_nestloop
where query = pg_last_query_id();
```

```
 query | slice | seg | step | duration | tasknum | rows  | tbl
-------+-------+-----+------+----------+---------+-------+-----
  6028 |     0 |   4 |    5 |       41 |      22 | 24277 | 240
  6028 |     1 |   4 |    5 |       26 |      23 | 24189 | 240
  6028 |     3 |   4 |    5 |       25 |      23 | 24376 | 240
  6028 |     2 |   4 |    5 |       54 |      22 | 23936 | 240
```

STL_PARSE

分析將字串剖析為二進位值以進行載入的查詢步驟。

所有使用者都可看見 STL_PARSE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_PARSE 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。

範例查詢

下列範例會傳回配量 1 和區段 0 的所有查詢步驟，其中字串已剖析為二進位值。

```
select query, step, starttime, endtime, tasknum, rows
from stl_parse
where slice=1 and segment=0;
```

```
 query | step |     starttime       |        endtime      | tasknum |  rows
-------+------+---------------------+---------------------+---------+--------
   669 |    1 | 2013-08-12 22:35:13 | 2013-08-12 22:35:17 |      32 | 192497
   696 |    1 | 2013-08-12 22:35:49 | 2013-08-12 22:35:49 |      32 |      0
   525 |    1 | 2013-08-12 22:32:03 | 2013-08-12 22:32:03 |      13 |  49990
   585 |    1 | 2013-08-12 22:33:18 | 2013-08-12 22:33:19 |      13 |    202
```

```
   621 |    1 | 2013-08-12 22:34:03 | 2013-08-12 22:34:03 |      27 |    365
   651 |    1 | 2013-08-12 22:34:47 | 2013-08-12 22:34:53 |      35 | 192497
   590 |    1 | 2013-08-12 22:33:28 | 2013-08-12 22:33:28 |      19 |      0
   599 |    1 | 2013-08-12 22:33:39 | 2013-08-12 22:33:39 |      31 |     11
   675 |    1 | 2013-08-12 22:35:26 | 2013-08-12 22:35:27 |      38 |   3766
   567 |    1 | 2013-08-12 22:32:47 | 2013-08-12 22:32:48 |      23 |  49990
   630 |    1 | 2013-08-12 22:34:17 | 2013-08-12 22:34:17 |      36 |      0
   572 |    1 | 2013-08-12 22:33:04 | 2013-08-12 22:33:04 |      29 |      0
   645 |    1 | 2013-08-12 22:34:37 | 2013-08-12 22:34:38 |      29 |   8798
   604 |    1 | 2013-08-12 22:33:47 | 2013-08-12 22:33:47 |      37 |      0
(14 rows)
```

STL_PLAN_INFO

使用 STL_PLAN_INFO 檢視，可以根據一組列查看查詢的 EXPLAIN 輸出。這是查看查詢計劃的其他方法。

所有使用者都可看見 STL_PLAN_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_PLAN_INFO 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別碼，其中節點會在查詢執行時映射至一個或多個步驟。
segment	integer	識別查詢區段的號碼。
step	integer	識別查詢步驟的號碼。

欄名稱	資料類型	描述
locus	integer	執行步驟的位置。若在運算節點上則為 0，若在領導者節點上則為 1。
plannode	integer	計劃節點的列舉值。請查看下表以取得 plannode 的列舉值。(STL_EXPLAIN 中的 PLANNODE 資料欄包含計劃節點文字。)
startupcost	double precision	傳回此步驟之第一列的預估相對成本。
totalcost	double precision	執行步驟的預估相對成本。
rows	bigint	步驟將產生之資料列的預估數目。
位元組	bigint	步驟將產生之位元組的預估數目。

範例查詢

下列範例會比較藉由使用 EXPLAIN 命令，以及藉由查詢 STL_PLAN_INFO 檢視所傳回之簡單 SELECT 查詢的查詢計劃。

```
explain select * from category;
QUERY PLAN
------------------------------------------------------------
XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)
(1 row)

select * from category;
catid | catgroup | catname | catdesc
-------+----------+-----------+--------------------------------------------
1 | Sports | MLB | Major League Baseball
3 | Sports | NFL | National Football League
5 | Sports | MLS | Major League Soccer
...

select * from stl_plan_info where query=256;

query | nodeid | segment | step | locus | plannode | startupcost | totalcost
| rows | bytes
```

```
-------+--------+---------+------+-------+----------+-------------+-----------+------
+-------
256 | 1 | 0 | 1 | 0 | 104 | 0 | 0.11 | 11 | 539
256 | 1 | 0 | 0 | 0 | 104 | 0 | 0.11 | 11 | 539
(2 rows)
```

在此範例中，PLANNODE 104 指的是 CATEGORY 資料表的循序掃描。

```
select distinct eventname from event order by 1;

eventname
----------------------------------------------------------------------------
.38 Special
3 Doors Down
70s Soul Jam
A Bronx Tale
...

explain select distinct eventname from event order by 1;

QUERY PLAN
------------------------------------------------------------------------------------------
XN Merge (cost=1000000000136.38..1000000000137.82 rows=576 width=17)
Merge Key: eventname
-> XN Network (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Send to leader
-> XN Sort (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Sort Key: eventname
-> XN Unique (cost=0.00..109.98 rows=576 width=17)
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=17)
(8 rows)

select * from stl_plan_info where query=240 order by nodeid desc;

query | nodeid | segment | step | locus | plannode | startupcost |
totalcost | rows | bytes
-------+--------+---------+------+-------+----------+------------------
+------------------+------+--------
240 | 5 | 0 | 0 | 0 | 104 | 0                | 87.98   | 8798 | 149566
240 | 5 | 0 | 1 | 0 | 104 | 0                | 87.98   | 8798 | 149566
```

```
240 | 4 | 0 | 2 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 0 | 3 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 0 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 1 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 3 | 1 | 2 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 3 | 2 | 0 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 2 | 2 | 1 | 0 | 123 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 1 | 3 | 0 | 0 | 122 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
(10 rows)
```

STL_PROJECT

包含用來評估表達式之查詢步驟的資料列。

所有使用者都可看見 STL_PROJECT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_PROJECT 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

欄名稱	資料類型	描述
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例會傳回查詢步驟的所有資料列，而這些查詢步驟是用來評估配量 0 和區段 1 的表達式。

```
select query, step, starttime, endtime, tasknum, rows
from stl_project
where slice=0 and segment=1;
```

```
 query  | step |         starttime   |          endtime    | tasknum | rows
--------+------+---------------------+---------------------+---------+------
  86399 |    2 | 2013-08-29 22:01:21 | 2013-08-29 22:01:21 |      25 |   -1
  86399 |    3 | 2013-08-29 22:01:21 | 2013-08-29 22:01:21 |      25 |   -1
    719 |    1 | 2013-08-12 22:38:33 | 2013-08-12 22:38:33 |       7 |   -1
  86383 |    1 | 2013-08-29 21:58:35 | 2013-08-29 21:58:35 |       7 |   -1
    714 |    1 | 2013-08-12 22:38:17 | 2013-08-12 22:38:17 |       2 |   -1
  86375 |    1 | 2013-08-29 21:57:59 | 2013-08-29 21:57:59 |       2 |   -1
  86397 |    2 | 2013-08-29 22:01:20 | 2013-08-29 22:01:20 |      19 |   -1
    627 |    1 | 2013-08-12 22:34:13 | 2013-08-12 22:34:13 |      34 |   -1
  86326 |    2 | 2013-08-29 21:45:28 | 2013-08-29 21:45:28 |      34 |   -1
  86326 |    3 | 2013-08-29 21:45:28 | 2013-08-29 21:45:28 |      34 |   -1
  86325 |    2 | 2013-08-29 21:45:27 | 2013-08-29 21:45:27 |      28 |   -1
  86371 |    1 | 2013-08-29 21:57:42 | 2013-08-29 21:57:42 |       4 |   -1
 111100 |    2 | 2013-09-03 19:04:45 | 2013-09-03 19:04:45 |      12 |   -1
    704 |    2 | 2013-08-12 22:36:34 | 2013-08-12 22:36:34 |      37 |   -1
    649 |    2 | 2013-08-12 22:34:47 | 2013-08-12 22:34:47 |      38 |   -1
    649 |    3 | 2013-08-12 22:34:47 | 2013-08-12 22:34:47 |      38 |   -1
    632 |    2 | 2013-08-12 22:34:22 | 2013-08-12 22:34:22 |      13 |   -1
    705 |    2 | 2013-08-12 22:36:48 | 2013-08-12 22:36:49 |      13 |   -1
    705 |    3 | 2013-08-12 22:36:48 | 2013-08-12 22:36:49 |      13 |   -1
```

```
      3 |    1 | 2013-08-12 20:07:40 | 2013-08-12 20:07:40 |       3 |   -1
  86373 |    1 | 2013-08-29 21:57:58 | 2013-08-29 21:57:58 |       3 |   -1
 107976 |    1 | 2013-09-03 04:05:12 | 2013-09-03 04:05:12 |       3 |   -1
  86381 |    1 | 2013-08-29 21:58:35 | 2013-08-29 21:58:35 |       8 |   -1
  86396 |    1 | 2013-08-29 22:01:20 | 2013-08-29 22:01:20 |      15 |   -1
    711 |    1 | 2013-08-12 22:37:10 | 2013-08-12 22:37:10 |      20 |   -1
  86324 |    1 | 2013-08-29 21:45:27 | 2013-08-29 21:45:27 |      24 |   -1
(26 rows)
```

STL_QUERY

傳回有關資料庫查詢的執行資訊。

> **Note**
>
> STL_QUERY 和 STL_QUERYTEXT 檢視僅包含查詢的相關資訊，不包含其他公用程式和 DDL 命令的相關資訊。如需 Amazon Redshift 執行之所有陳述式的清單和相關資訊，您也可以查詢 STL_DDLTEXT 和 STL_UTILITYTEXT 檢視。如需 Amazon Redshift 執行之所有陳述式的完整清單，您可以查詢 SVL_STATEMENTTEXT 檢視。

所有使用者都可看見 STL_QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位值為 `default`。
xid	bigint	交易 ID。

欄名稱	資料類型	描述
pid	integer	處理程序 ID。正常情況下，工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。在特定內部事件後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 PID。如需詳細資訊，請參閱 STL_RESTARTED_SESSIONS。
database	character(32)	當查詢發出時，要將使用者連接至其中的資料庫名稱。
querytxt	character(4000)	查詢的實際查詢文字。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 **1**。如果查詢完成執行 (包括將結果傳回至用戶端)，則此資料欄包含 **0**。如果用戶端在接收結果之前中斷連接，則查詢將標示為已取消 (**1**)，即使它已在後端成功完成也一樣。
insert_pristine	integer	當目前查詢正在執行時，寫入查詢是否能夠執行。1 = 不允許寫入查詢。0 = 允許寫入查詢。此欄位適用於偵錯。
concurrency_scaling_status	integer	指出查詢是執行於主要叢集或並行擴展叢集。可能的值如下： 0 - 執行於主要叢集 1 - 執行於並行擴展叢集 大於 1 - 執行於主要叢集

範例查詢

下列查詢會列出五個最新查詢。

```
select query, trim(querytxt) as sqlquery
from stl_query
order by query desc limit 5;

query |                                  sqlquery
------+--------------------------------------------------
129 | select query, trim(querytxt) from stl_query order by query;
128 | select node from stv_disk_read_speeds;
127 | select system_status from stv_gui_status
126 | select * from systable_topology order by slice
125 | load global dict registry
(5 rows)
```

下列查詢會針對 2013 年 2 月 15 日執行的查詢，依遞減順序傳回經過時間。

```
select query, datediff(seconds, starttime, endtime),
trim(querytxt) as sqlquery
from stl_query
where starttime >= '2013-02-15 00:00' and endtime < '2013-02-16 00:00'
order by date_diff desc;

 query | date_diff |  sqlquery
-------+-----------+-------------------------------------------
 55    |       119 | padb_fetch_sample: select count(*) from category
121    |         9 | select * from svl_query_summary;
181    |         6 | select * from svl_query_summary where query in(179,178);
172    |         5 | select * from svl_query_summary where query=148;
...
(189 rows)
```

下列查詢顯示查詢的佇列時間和執行時間。concurrency_scaling_status = 1 的查詢執行於並行擴展叢集。所有其他查詢皆執行於主要叢集。

```
SELECT w.service_class AS queue
     , q.concurrency_scaling_status
     , COUNT( * ) AS queries
     , SUM( q.aborted )  AS aborted
     , SUM( ROUND( total_queue_time::NUMERIC / 1000000,2 ) ) AS queue_secs
```

```
     , SUM( ROUND( total_exec_time::NUMERIC / 1000000,2 ) )  AS exec_secs
FROM stl_query q
     JOIN stl_wlm_query w
          USING (userid,query)
WHERE q.userid > 1
  AND service_class > 5
  AND q.starttime > '2019-03-01 16:38:00'
  AND q.endtime   < '2019-03-01 17:40:00'
GROUP BY 1,2
ORDER BY 1,2;
```

STL_QUERY_METRICS

包含已在使用者定義的查詢佇列 (服務類別) 中完成執行之查詢的指標資訊，例如已處理的資料列數目、CPU 用量、輸入/輸出和磁碟使用情形。若要檢視目前執行中之作用中查詢的指標，請參閱 STV_QUERY_METRICS 系統檢視。

查詢指標每間隔一秒鐘取樣一次。因此，相同查詢的不同執行可能會傳回稍微不同的時間。另外，可能不會記錄執行時間不到一秒的查詢區段。

STL_QUERY_METRICS 會追蹤並彙總查詢、區段和步驟層級的指標。如需查詢區段和步驟的相關資訊，請參閱查詢計劃和執行工作流程。許多指標 (例如 max_rows、cpu_time 等等) 是跨節點配量加總的。如需節點配量的相關資訊，請參閱資料倉儲系統架構。

若要判斷資料列報告指標的層級，請檢查 segment 和 step_type 資料欄。

- 如果 segment 和 step_type 皆為 -1，則資料列報告查詢層級的指標。
- 如果 segment 不是 -1，且 step_type 是 -1，則資料列報告區段層級的指標。
- 如果 segment 和 step_type 皆不是 -1，則資料列報告步驟層級的指標。

SVL_QUERY_METRICS 檢視和 SVL_QUERY_METRICS_SUMMARY 檢視彙總此檢視中的資料，並以更易存取的形式呈現資訊。

所有使用者都可看見 STL_QUERY_METRICS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
service_class	integer	服務類別的 ID。查詢佇列定義於 WLM 組態。只會回報使用者定義之佇列的指標。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。如果區段值為 -1，指標區段值將累計至查詢層級。
step_type	integer	執行的步驟類型。如需步驟類型的說明，請參閱步驟類型。
starttime	timestamp	查詢開始執行的 UTC 時間，精確度為 6 位數的小數秒。例如：`2009-06-12 11:29:19.131358` 。
slices	integer	叢集的分割數量。
max_rows	bigint	為步驟匯出的最大資料列數 (彙總所有分割)。
rows	bigint	步驟處理的資料列數。
max_cpu_time	bigint	已使用的最大 CPU 時間，以微秒為單位。在區段層級，區段中所有分割已使用的最大 CPU 時間。在查詢層級，任何查詢區段已使用的最大 CPU 時間。
cpu_time	bigint	已使用的 CPU 時間，以微秒為單位。在區段層級，所有分割的區段所使用的 CPU 總時間。在查詢層級，所有分割與區段的查詢 CPU 時間總和。
max_blocks_read	bigint	區段讀取的 1 MB 區塊最大數量 (彙總所有分割)。在區段層級，所有分割的區段所讀取的 1 MB 區塊最大數量。在查詢層級，任何查詢區段所讀取的 1 MB 區塊最大數量。
blocks_read	bigint	查詢或區段所讀取的 1 MB 區塊數。

欄名稱	資料類型	描述
max_run_time	bigint	區段的最大經過時間 (以微秒為單位)。在區段層級，所有分割之區段的最大執行時間。在查詢層級，任何查詢區段的最大執行時間。
run_time	bigint	所有分割合計的總執行時間。執行時間不包含等待時間。 在區段層級，所有分割合計的區段執行時間。在查詢層級，所有分割與區段的查詢執行時間總和。因為此值為總和，因此執行時間與查詢執行時間無關。
max_blocks_to_disk	bigint	用來寫入中繼結果之磁碟空間的數量上限，以 MB 區塊表示。在區段層級，所有分割之區段已使用的最大磁碟空間量。在查詢層級，任何查詢區段已使用的最大磁碟空間量。
blocks_to_disk	bigint	查詢或區段用來寫入中繼結果之磁碟空間的數量上限，以 MB 區塊表示。
step	integer	執行的查詢步驟。
max_query_scan_size	bigint	查詢所掃描的最大資料大小，以 MB 為單位。在區段層級，所有分割之區段所掃描的最大資料大小。在查詢層級，任何查詢區段所掃描的最大資料大小。
query_scan_size	bigint	查詢所掃描的資料大小，以 MB 為單位。
query_priority	integer	查詢的優先順序。可能的值為 -1、0、1、2、3 和 4，其中 -1 表示不支援查詢優先順序。
query_queue_time	bigint	查詢排入佇列的時間 (毫秒)。
service_class_name	character (64)	服務類別的名稱。

範例查詢

若要找出具有高 CPU 時間 (超過 1,000 秒) 的查詢，請執行下列查詢。

```
Select query, cpu_time / 1000000 as cpu_seconds
from stl_query_metrics where segment = -1 and cpu_time > 1000000000
order by cpu_time;

query | cpu_seconds
------+------------
25775 |        9540
```

若要找出具有傳回超過一百萬個資料列之巢狀迴圈聯結的使用中查詢，請執行下列查詢。

```
select query, rows
from stl_query_metrics
where step_type = 15 and rows > 1000000
order by rows;

query | rows
------+-----------
25775 | 2621562702
```

若要找出已執行超過 60 秒，但已使用 CPU 時間少於 10 秒的使用中查詢，請執行下列查詢。

```
select query, run_time/1000000 as run_time_seconds
from stl_query_metrics
where segment = -1 and run_time > 60000000 and cpu_time < 10000000;

query | run_time_seconds
------+-----------------
25775 |              114
```

STL_QUERYTEXT

擷取 SQL 命令的查詢文字。

查詢 STL_QUERYTEXT 檢視，以擷取針對下列陳述式記錄的 SQL：

- SELECT、SELECT INTO
- INSERT、UPDATE、DELETE
- COPY
- UNLOAD

- 透過執行 VACUUM 和 ANALYZE 產生的查詢
- CREATE TABLE AS (CTAS)

若要查詢這些陳述式在特定時段的活動，請聯結 STL_QUERYTEXT 和 STL_QUERY 檢視。

> **Note**
>
> STL_QUERY 和 STL_QUERYTEXT 檢視僅包含查詢的相關資訊，不包含其他公用程式和 DDL 命令的相關資訊。如需 Amazon Redshift 執行之所有陳述式的清單和相關資訊，您也可以查詢 STL_DDLTEXT 和 STL_UTILITYTEXT 檢視。如需 Amazon Redshift 執行之所有陳述式的完整清單，您可以查詢 SVL_STATEMENTTEXT 檢視。

另請參閱 STL_DDLTEXT、STL_UTILITYTEXT 和 SVL_STATEMENTTEXT。

所有使用者都可看見 STL_QUERYTEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_TEXT 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。正常情況下，工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。在特定內部事件後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 PID。如需詳細資訊，請參閱 STL_RESTARTED_SESSIONS。您可以使用此欄來聯結至 STL_ERROR 檢視。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。

欄名稱	資料類型	描述
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\\) 和換行符號 (\n) 等特殊字元。

範例查詢

您可以使用 PG_BACKEND_PID() 函數，來擷取目前工作階段的資訊。例如，下列查詢會針對目前工作階段中完成的查詢傳回其查詢 ID 和一部分的查詢文字。

```
select query, substring(text,1,60)
from stl_querytext
where pid = pg_backend_pid()
order by query desc;

 query |                          substring
-------+------------------------------------------------------------
 28262 | select query, substring(text,1,80) from stl_querytext where
 28252 | select query, substring(path,0,80) as path from stl_unload_l
 28248 | copy category from 's3://dw-tickit/manifest/category/1030_ma
 28247 | Count rows in target table
 28245 | unload ('select * from category') to 's3://dw-tickit/manifes
 28240 | select query, substring(text,1,40) from stl_querytext where
(6 rows)
```

重建儲存的 SQL

若要重建儲存在 STL_QUERYTEXT `text` 資料欄中的 SQL，則需執行 SELECT 陳述式以從 `text` 資料欄的一或多個部分建立 SQL。請先以新的一行取代任意 (\n) 特殊字元，再執行重建的 SQL。下列 SELECT 陳述式顯示的結果會是 `query_statement` 欄中重建的 SQL 資料列。

```
SELECT query, LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END)
 WITHIN GROUP (ORDER BY sequence) as query_statement, COUNT(*) as row_count
FROM stl_querytext GROUP BY query ORDER BY query desc;
```

例如，下列查詢會選取 3 個資料欄。該查詢本身的長度超過 200 個字元，且會儲存在 STL_QUERYTEXT 的多個部分中。

```
select
1 AS a0123456789012345678901234567890123456789012345678901234567890,
2 AS b0123456789012345678901234567890123456789012345678901234567890,
3 AS b012345678901234567890123456789012345678901234
FROM stl_querytext;
```

在本範例中，查詢會儲存在 STL_QUERYTEXT `text` 資料欄的 2 個部分 (資料列) 中。

```
select query, sequence, text
from stl_querytext where query=pg_last_query_id() order by query desc, sequence limit
 10;
```

```

query  | sequence |
                         text

-------+----------
+-------------------------------------------------------------------------------------------
    45 |        0 | select\n1 AS
 a0123456789012345678901234567890123456789012345678901234567890,\n2 AS
 b0123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b012345678901234567890123456789012345678901234
    45 |        1 | \nFROM stl_querytext;
```

若要重建儲存在 STL_QUERYTEXT 中的 SQL，請執行下列 SQL。

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS text
from stl_querytext where query=pg_last_query_id();
```

若要在用戶端中使用產生的重建 SQL，請以新的一行取代任意 (\n) 特殊字元。

```


                      text

--------------------------------------------------------------------------------------------
```

```
 select\n1 AS a01234567890123456789012345678901234567890123456789012345678901234567890,
\n2 AS b01234567890123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b01234567890123456789012345678901234567890123456789012345678901234\nFROM stl_querytext;
```

STL_REPLACEMENTS

顯示一個日誌，其記錄搭配 ACCEPTINVCHARS 選項的 COPY 命令何時取代無效的 UTF-8 字元。在至少需要一個取代項目的每個節點上，對於其前 100 個資料列的每一個都會新增一個日誌項目至 STL_REPLACEMENTS。

所有使用者都可看見 STL_REPLACEMENTS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_NESTLOOP 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_COPY_REPLACEMENTS。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	發生取代的節點配量號碼。
tbl	integer	表格 ID。
starttime	timestamp	COPY 命令的開始時間，以 UTC 表示。
session	integer	執行 COPY 命令之工作階段的工作階段 ID。
filename	character (256)	用於 COPY 命令之輸入檔案的完整路徑。

欄名稱	資料類型	描述
line_number	bigint	輸入資料檔案中包含無效 UTF-8 字元的行號。-1 表示行號不可用，例如從單欄式資料檔案複製時。
colname	character (127)	包含無效 UTF-8 字元的第一個欄位。
raw_line	character (1024)	包含無效 UTF-8 字元的原始載入資料。

範例查詢

下列範例會傳回最新 COPY 操作的取代項目。

```
select query, session, filename, line_number, colname
from stl_replacements
where query = pg_last_copy_id();

 query | session |   filename                          | line_number | colname
 ------+---------+-------------------------------------+-------------+--------
    96 |    6314 | s3://mybucket/allusers_pipe.txt   |         251 | city
    96 |    6314 | s3://mybucket/allusers_pipe.txt   |         317 | city
    96 |    6314 | s3://mybucket/allusers_pipe.txt   |         569 | city
    96 |    6314 | s3://mybucket/allusers_pipe.txt   |         623 | city
    96 |    6314 | s3://mybucket/allusers_pipe.txt   |         694 | city
...
```

STL_RESTARTED_SESSIONS

為了要在特定內部事件後維持持續可用性，Amazon Redshift 可能會利用新的處理程序 ID (PID) 重新啟動作用中工作階段。當 Amazon Redshift 重新啟動工作階段時，STL_RESTARTED_SESSIONS 會記錄新的 PID 和舊的 PID。

如需詳細資訊，請參閱本節中的下列範例。

所有使用者都可看見 STL_RESTARTED_SESSIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_SESSION_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
currenttime	timestamp	事件的時間。
dbname	character (50)	與工作階段相關聯之資料庫的名稱。
newpid	integer	已重新啟動之工作階段的處理程序 ID。
oldpid	integer	原始工作階段的處理程序 ID。
使用者名稱	character (50)	與工作階段相關聯之使用者的名稱。
remotehost	character (45)	遠端主機的名稱或 IP 地址。
remoteport	character (32)	遠端主機的連接埠號碼。
parkedtime	timestamp	此資訊僅供內部使用。
session_vars	character (2000)	此資訊僅供內部使用。

範例查詢

下列範例聯結 STL_RESTARTED_SESSIONS 與 STL_SESSIONS，為已重新啟動的工作階段顯示使用者名稱。

```
select process, stl_restarted_sessions.newpid, user_name
from stl_sessions
inner join stl_restarted_sessions on stl_sessions.process =
 stl_restarted_sessions.oldpid
order by process;

...
```

STL_RETURN

包含查詢中傳回步驟的詳細資訊。傳回步驟會將運算節點上完成之查詢的結果傳回至領導者節點。然後，領導者節點會合併資料，並將結果傳回至提出請求的用戶端。對於領導者節點上完成的查詢，傳回步驟會將結果傳回至用戶端。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱 查詢處理。

所有使用者都可看見 STL_RETURN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_RETURN 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

欄名稱	資料類型	描述
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
packets	integer	透過網路傳送的封包總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例顯示最新查詢中的哪些步驟已在每個配量上執行。

```
SELECT query, slice, segment, step, endtime, rows, packets
from stl_return where query = pg_last_query_id();

 query |  slice | segment | step |          endtime           | rows | packets
-------+--------+---------+------+----------------------------+------+---------
     4 |      2 |       3 |    2 | 2013-12-27 01:43:21.469043 |    3 |       0
     4 |      3 |       3 |    2 | 2013-12-27 01:43:21.473321 |    0 |       0
     4 |      0 |       3 |    2 | 2013-12-27 01:43:21.469118 |    2 |       0
     4 |      1 |       3 |    2 | 2013-12-27 01:43:21.474196 |    0 |       0
     4 |      4 |       3 |    2 | 2013-12-27 01:43:21.47704  |    2 |       0
     4 |      5 |       3 |    2 | 2013-12-27 01:43:21.478593 |    0 |       0
     4 |   12811|       4 |    1 | 2013-12-27 01:43:21.480755 |    0 |       0
(7 rows)
```

STL_S3CLIENT

記錄傳輸時間和其他效能指標。

使用 STL_S3CLIENT 資料表來尋找從 Amazon S3 傳輸資料所花費的時間。

所有使用者都可看見 STL_S3CLIENT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
recordtime	timestamp	記錄被記錄的時間。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。
http_method	character (64)	對應至 Amazon S3 請求的 HTTP 方法名稱。
儲存貯體	character (64)	S3 儲存貯體名稱。
金鑰	character (256)	對應至 Amazon S3 物件的索引鍵。
transfer_size	bigint	已傳輸的位元組數目。
data_size	bigint	資料的位元組數目。此值與未壓縮資料的 transfer_size 相同。如果已使用壓縮，則這是未壓縮資料的大小。
start_time	bigint	自 2000 年 1 月 1 日起，傳輸開始的時間 (以微秒為單位)。
end_time	bigint	自 2000 年 1 月 1 日起，傳輸結束的時間 (以微秒為單位)。
transfer_time	bigint	傳輸所花費的時間 (以微秒為單位)。
compression_time	bigint	傳輸時間中未壓縮資料所花費的部分 (以微秒為單位)。
connect_time	bigint	從開始直到完成連線至遠端伺服器的時間 (以微秒為單位)。

欄名稱	資料類型	描述
app_connect_time	bigint	從開始直到完成與遠端主機的 SSL 連接/交握的時間 (以微秒為單位)。
retries	bigint	已嘗試傳輸的次數。
request_id	char(32)	來自 Amazon S3 HTTP 回應標題的請求 ID
extended_request_id	char(128)	來自 Amazon S3 HTTP 標題回應的延伸請求 ID (x-amz-id-2)。
ip_address	char(64)	伺服器的 IP 地址 (ip V4 或 V6)。
is_partial	integer	如果值為 true (1)，則表示輸入檔在 COPY 操作期間被分割為多個範圍。如果此值為 false (0)，則輸入檔案不會分割。
start_offset	bigint	如果輸入檔案在 COPY 操作期間被分割，則該值指示分割的偏移值 (以位元組為單位)。如果檔案未分割，則此值為 0。

範例查詢

下列查詢會傳回使用 COPY 命令來載入檔案所花費的時間。

```
select slice, key, transfer_time
from stl_s3client
where query = pg_last_copy_id();
```

結果

```
 slice |   key                          | transfer_time
------+--------------------------------+---------------
     0 | listing10M0003_part_00         |    16626716
     1 | listing10M0001_part_00         |    12894494
     2 | listing10M0002_part_00         |    14320978
     3 | listing10M0000_part_00         |    11293439
  3371 | prefix=listing10M;marker=      |       99395
```

下列查詢會將 `start_time` 和 `end_time` 轉換為時間戳記。

```
select userid,query,slice,pid,recordtime,start_time,end_time,
'2000-01-01'::timestamp + (start_time/1000000.0)* interval '1 second' as start_ts,
'2000-01-01'::timestamp + (end_time/1000000.0)* interval '1 second' as end_ts
from stl_s3client where query> -1 limit 5;
```

```
 userid | query | slice |  pid  |         recordtime         |   start_time    |
 end_time     |          start_ts          |           end_ts
--------+-------+-------+-------+----------------------------+-----------------
+-----------------+----------------------------+----------------------------
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.207839 | 616436837154256 |
 616436837207838 | 2019-07-14 16:27:17.154256 | 2019-07-14 16:27:17.207838
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.252521 | 616436837208208 |
 616436837252520 | 2019-07-14 16:27:17.208208 | 2019-07-14 16:27:17.25252
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.284376 | 616436837208460 |
 616436837284374 | 2019-07-14 16:27:17.20846  | 2019-07-14 16:27:17.284374
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.285307 | 616436837208980 |
 616436837285306 | 2019-07-14 16:27:17.20898  | 2019-07-14 16:27:17.285306
      0 |     0 |     0 | 23449 | 2019-07-14 16:27:17.353853 | 616436837302216 |
 616436837353851 | 2019-07-14 16:27:17.302216 | 2019-07-14 16:27:17.353851
```

STL_S3CLIENT_ERROR

記錄從 Amazon S3 載入檔案時配量遇到的錯誤。

使用 STL_S3CLIENT_ERROR 來尋找從 Amazon S3 傳輸資料時所遇到之錯誤的詳細資訊，做為 COPY 命令的一部分。

所有使用者都可看見 STL_S3CLIENT_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。查詢 ID -1 供內部使用。

欄名稱	資料類型	描述
sliceid	integer	識別執行查詢之配量的數字。
recordtime	timestamp	記錄被記錄的時間。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。
http_method	character (64)	對應至 Amazon S3 請求的 HTTP 方法名稱。
儲存貯體	character (64)	Amazon S3 儲存貯體名稱。
金鑰	character (256)	對應至 Amazon S3 物件的索引鍵。
error	character (1024)	錯誤訊息。
is_partial	integer	如果值為 true (1)，則表示輸入檔在 COPY 操作期間被分割為多個範圍。如果此值為 false (0)，則輸入檔案不會分割。
start_offset	bigint	如果輸入檔案在 COPY 操作期間被分割，則該值指示分割的偏移值 (以位元組為單位)。如果檔案未分割，則此值為 0。

使用須知

如果看到多個錯誤具有「連線逾時」，則可能發生聯網問題。如果您使用的是增強型 VPC 路由，請驗證您在叢集 VPC 與資料來源之間具有一個有效的網路路徑。如需詳細資訊，請參閱 Amazon Redshift 增強型 VPC 路由。

範例查詢

下列查詢會從目前工作階段期間完成的 COPY 命令傳回錯誤。

```
select query, sliceid, substring(key from 1 for 20) as file,
substring(error from 1 for 35) as error
```

```
from stl_s3client_error
where pid = pg_backend_pid()
order by query desc;
```

結果

```
 query  | sliceid |     file          |             error

--------+---------+-------------------+-----------------------------------
 362228 |      12 | part.tbl.25.159.gz | transfer closed with 1947655 bytes
 362228 |      24 | part.tbl.15.577.gz | transfer closed with 1881910 bytes
 362228 |       7 | part.tbl.22.600.gz | transfer closed with 700143 bytes r
 362228 |      22 | part.tbl.3.34.gz   | transfer closed with 2334528 bytes
 362228 |      11 | part.tbl.30.274.gz | transfer closed with 699031 bytes r
 362228 |      30 | part.tbl.5.509.gz  | Unknown SSL protocol error in conne
 361999 |      10 | part.tbl.23.305.gz | transfer closed with 698959 bytes r
 361999 |      19 | part.tbl.26.582.gz | transfer closed with 1881458 bytes
 361999 |       4 | part.tbl.15.629.gz | transfer closed with 2275907 bytes
 361999 |      20 | part.tbl.6.456.gz  | transfer closed with 692162 bytes r
(10 rows)
```

STL_SAVE

包含查詢中儲存步驟的詳細資訊。儲存步驟會將輸入串流儲存至暫時性資料表。暫時性資料表是暫存資料表，其會在查詢執行期間儲存中繼結果。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱 查詢處理。

所有使用者都可看見 STL_SAVE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_SAVE 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
tbl	integer	具體化的暫時性資料表的 ID。
is_diskbased	character(1)	查詢的這個步驟是否以磁碟型操作方式執行：true (t) 或 false (f)。
workmem	bigint	已指派給步驟之運作中記憶體的位元組數。

範例查詢

下列範例顯示最新查詢中的哪些儲存步驟已在每個配量上執行。

```
select query, slice, segment, step, tasknum, rows,  tbl
```

```
from stl_save where query = pg_last_query_id();

 query | slice | segment | step | tasknum | rows | tbl
-------+-------+---------+------+---------+------+-----
 52236 |     3 |       0 |    2 |      21 |    0 | 239
 52236 |     2 |       0 |    2 |      20 |    0 | 239
 52236 |     2 |       2 |    2 |      20 |    0 | 239
 52236 |     3 |       2 |    2 |      21 |    0 | 239
 52236 |     1 |       0 |    2 |      21 |    0 | 239
 52236 |     0 |       0 |    2 |      20 |    0 | 239
 52236 |     0 |       2 |    2 |      20 |    0 | 239
 52236 |     1 |       2 |    2 |      21 |    0 | 239
(8 rows)
```

STL_SCAN

分析查詢的資料表掃描步驟。此資料表中資料列的步驟號碼一律為 0，因為掃描是區段中的第一個步驟。

所有使用者都可看見 STL_SCAN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_SCAN 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。

欄名稱	資料類型	描述
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
fetches	bigint	此資訊僅供內部使用。
type	integer	掃描類型的 ID。如需有效值的清單，請參閱下列資料表。
tbl	integer	表格 ID。
is_rrscan	character(1)	若為 true (t)，表示已在步驟上使用範圍限制掃描。
is_delayed_scan	character(1)	此資訊僅供內部使用。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之後，但在套用使用者定義的查詢篩選條件之前已發出的資料列總數。
rows_pre_user_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之後，但在套用使用者定義的查詢篩選條件之前已處理的資料列數目。
perm_table_name	character(136)	對於永久資料表的掃描，指的是已掃描的資料表名稱。

欄名稱	資料類型	描述
is_rlf_scan	character(1)	若為 true (t)，表示已在步驟上使用資料列層級篩選。
is_rlf_scan_reason	integer	此資訊僅供內部使用。
num_em_blocks	integer	此資訊僅供內部使用。
checksum	bigint	此資訊僅供內部使用。
runtime_filtering	character(1)	如果為 true (t)，則表示會套用執行時間篩選條件。
scan_region	integer	此資訊僅供內部使用。
num_sortkey_as_predicate	integer	此資訊僅供內部使用。
row_fetcher_state	integer	此資訊僅供內部使用。
consumed_scan_ranges	bigint	此資訊僅供內部使用。
work_stealing_reason	bigint	此資訊僅供內部使用。
is_vectorized_scan	character(1)	此資訊僅供內部使用。
is_vectorized_scan_reason	integer	此資訊僅供內部使用。

欄名稱	資料類型	描述
row_fetcher_reason	bigint	此資訊僅供內部使用。
topology_signature	bigint	此資訊僅供內部使用。
use_tpm_partition	character(1)	此資訊僅供內部使用。
is_rrscan_expr	character(1)	此資訊僅供內部使用。
scanned_mega_value	character(1)	此資訊僅供內部使用。此資訊顯示指定的掃描步驟是否掃描了較大的值。大值將儲存在多個區塊中。區塊大小預設為 1 MB，在預設設定中，大值大於 1 MB。

掃描類型

類型 ID	描述
1	來自網路的資料。
2	已壓縮之共用記憶體中的永久使用者資料表。
3	臨時性全資料列資料表。
21	從 Amazon S3 載入檔案。
22	從 Amazon DynamoDB 載入資料表。
23	從遠端 SSH 連線載入資料。
24	從遠端叢集載入資料 (已排序區域)。這是用於調整大小。
25	從遠端叢集載入資料 (未排序區域)。這是用於調整大小。
28	使用 UNION ALL 在多個資料表上從時間序列檢視中讀取資料。

類型 ID	描述
29	從 Amazon S3 外部資料表讀取資料。
30	讀取 Amazon S3 外部資料表的分割區資訊。
33	從遠端 Postgres 資料表讀取資料。
36	從遠端 MySQL 資料表讀取資料。
37	從遠端 Kinesis 串流讀取資料。

使用須知

在理想情況下，rows 應該相當接近 rows_pre_filter。rows 與 rows_pre_filter 之間的差異若很大，表示執行引擎正在掃描稍後將捨棄的資料列，這樣做缺乏效率。rows_pre_filter 與 rows_pre_user_filter 之間的差異是掃描中幽靈資料列的數目。執行 VACUUM 以移除標記進行刪除的資料。rows 與 rows_pre_user_filter 之間的差異是查詢所篩選的資料列數目。若有許多資料列已遭使用者篩選條件捨棄，請檢閱您選擇的排序資料欄，或若這是由於大型未排序區域所致，請執行清空。

範例查詢

下列範例顯示 rows_pre_filter 大於 rows_pre_user_filter，因為資料表已刪除未清空的資料列 (幽靈資料列)。

```
SELECT query, slice, segment,step,rows, rows_pre_filter, rows_pre_user_filter
from stl_scan where query = pg_last_query_id();

 query |  slice | segment | step | rows  | rows_pre_filter | rows_pre_user_filter
-------+--------+---------+------+-------+-----------------+----------------------
 42915 |      0 |       0 |    0 | 43159 |           86318 |                43159
 42915 |      0 |       1 |    0 |     1 |               0 |                    0
 42915 |      1 |       0 |    0 | 43091 |           86182 |                43091
 42915 |      1 |       1 |    0 |     1 |               0 |                    0
 42915 |      2 |       0 |    0 | 42778 |           85556 |                42778
 42915 |      2 |       1 |    0 |     1 |               0 |                    0
 42915 |      3 |       0 |    0 | 43428 |           86856 |                43428
 42915 |      3 |       1 |    0 |     1 |               0 |                    0
 42915 |  10000 |       2 |    0 |     4 |               0 |                    0
(9 rows)
```

STL_SCHEMA_QUOTA_VIOLATIONS

超過結構描述配額時，記錄出現次數、時間戳記、XID 和其他有用的資訊。

所有使用者都可看見 STL_SCHEMA_QUOTA_VIOLATIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_SCHEMA_QUOTA_VIOLATIONS 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	Description
ownerid	integer	結構描述擁有者的 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。
userid	integer	產生項目的使用者之 ID。
schema_id	integer	命名空間或結構描述 ID。
schema_name	character (128)	命名空間或結構描述名稱。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。
disk_usage_pct	double precision	結構描述目前從所設定配額使用的磁碟空間百分比。
timestamp	沒有時區的時間戳記	發生違規的時間。

範例查詢

下列查詢顯示配額違規的結果：

```
SELECT userid, TRIM(SCHEMA_NAME) "schema_name", quota, disk_usage, disk_usage_pct,
 timestamp FROM
stl_schema_quota_violations WHERE SCHEMA_NAME = 'sales_schema' ORDER BY timestamp DESC;
```

此查詢會傳回所指定結構描述的下列範例輸出：

```
userid | schema_name  | quota | disk_usage | disk_usage_pct |timestamp
-------+--------------+-------+------------+----------------
+----------------------------
104    | sales_schema | 2048  | 2798       |  136.62        | 2020-04-20
 20:09:25.494723
(1 row)
```

STL_SESSIONS

傳回使用者工作階段歷史記錄的相關資訊。

STL_SESSIONS 不同於 STV_SESSIONS，其中 STL_SESSIONS 包含工作階段歷史記錄，而 STV_SESSIONS 包含目前作用中的工作階段。

所有使用者都可看見 STL_SESSIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_SESSION_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
starttime	timestamp	工作階段開始的時間，以 UTC 表示。
endtime	timestamp	工作階段結束的時間，以 UTC 表示。
process	integer	工作階段的處理程序 ID。

欄名稱	資料類型	描述
user_name	character(50)	與工作階段相關聯的使用者名稱。
db_name	character(50)	與工作階段相關聯之資料庫的名稱。
timeout_sec	int	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。
timed_out	int	指示工作階段是否逾時的值：如果逾時，則為 1，否則為 0。

範例查詢

若要檢視 TICKIT 資料庫的工作階段歷史記錄，請輸入下列查詢：

```
select starttime, process, user_name, timeout_sec, timed_out
from stl_sessions
where db_name='tickit' order by starttime;
```

此查詢傳回下列範例輸出：

```
    starttime              | process |  user_name              | timeout_sec | timed_out
---------------------------+---------+-------------------------+-------------
+-------------
2008-09-15 09:54:06.746705 |   32358 | dwuser                  | 120         | 1
2008-09-15 09:56:34.30275  |   32744 | dwuser                  | 60          | 1
2008-09-15 11:20:34.694837 |   14906 | dwuser                  | 0           | 0
2008-09-15 11:22:16.749818 |   15148 | dwuser                  | 0           | 0
2008-09-15 14:32:44.66112  |   14031 | dwuser                  | 0           | 0
2008-09-15 14:56:30.22161  |   18380 | dwuser                  | 0           | 0
2008-09-15 15:28:32.509354 |   24344 | dwuser                  | 0           | 0
2008-09-15 16:01:00.557326 |   30153 | dwuser                  | 120         | 1
2008-09-15 17:28:21.419858 |   12805 | dwuser                  | 0           | 0
2008-09-15 20:58:37.601937 |   14951 | dwuser                  | 60          | 1
2008-09-16 11:12:30.960564 |   27437 | dwuser                  | 60          | 1
2008-09-16 14:11:37.639092 |   23790 | dwuser                  | 3600        | 1
2008-09-16 15:13:46.02195  |    1355 | dwuser                  | 120         | 1
2008-09-16 15:22:36.515106 |    2878 | dwuser                  | 120         | 1
2008-09-16 15:44:39.194579 |    6470 | dwuser                  | 120         | 1
```

```
2008-09-16 16:50:27.02138  |   17254 | dwuser                | 120          | 1
2008-09-17 12:05:02.157208 |    8439 | dwuser                | 3600         | 0
(17 rows)
```

STL_SORT

顯示查詢的排序執行步驟，例如使用 ORDER BY 處理的步驟。

所有使用者都可看見 STL_SORT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_SORT 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

欄名稱	資料類型	描述
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
tbl	integer	表格 ID。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體中的位元組總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例傳回配量 0 和區段 1 的排序結果。

```
select query, bytes, tbl, is_diskbased, workmem
from stl_sort
where slice=0 and segment=1;
```

```
 query |  bytes  | tbl | is_diskbased |  workmem
-------+---------+-----+--------------+-----------
   567 | 3126968 | 241 | f            | 383385600
   604 |    5292 | 242 | f            | 383385600
   675 |  104776 | 251 | f            | 383385600
   525 | 3126968 | 251 | f            | 383385600
   585 |    5068 | 241 | f            | 383385600
   630 |  204808 | 266 | f            | 383385600
   704 |       0 | 242 | f            |         0
   669 | 4606416 | 241 | f            | 383385600
   696 |  104776 | 241 | f            | 383385600
   651 | 4606416 | 254 | f            | 383385600
   632 |       0 | 256 | f            |         0
   599 |     396 | 241 | f            | 383385600
 86397 |       0 | 242 | f            |         0
   621 |    5292 | 241 | f            | 383385600
```

```
 86325 |       0 | 242 | f            |         0
   572 |    5068 | 242 | f            | 383385600
   645 |  204808 | 241 | f            | 383385600
   590 |     396 | 242 | f            | 383385600
(18 rows)
```

STL_SSHCLIENT_ERROR

記錄 SSH 用戶端看到的所有錯誤。

所有使用者都可看見 STL_SSHCLIENT_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
recordtime	timestamp	記錄錯誤的時間。
pid	integer	記錄錯誤的處理程序。
ssh_username	character (1024)	SSH 使用者名稱。
端點	character (1024)	SSH 端點。
command	character (4096)	完整 SSH 命令。
error	character (1024)	錯誤訊息。

STL_STREAM_SEGS

列出串流與並行區段之間的關係。

此內容中的串流是 Amazon Redshift 串流。此系統檢視不屬於 串流擷取。

所有使用者都可看見 STL_STREAM_SEGS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_STREAM_SEGS 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
串流	integer	查詢的並行區段集。
segment	integer	識別查詢區段的號碼。

範例查詢

若要針對最新查詢檢視串流與並行區段之間的關係，請輸入下列查詢：

```
select *
from stl_stream_segs
where query = pg_last_query_id();

 query | stream | segment
-------+--------+---------
    10 |      1 |       2
```

```
    10 |      0 |       0
    10 |      2 |       4
    10 |      1 |       3
    10 |      0 |       1
(5 rows)
```

STL_TR_CONFLICT

顯示資訊以識別並解決交易與資料庫資料表的衝突。

當兩個以上使用者正在查詢和修改資料表中的資料列，以致其交易無法序列化時，即會發生交易衝突。交易若執行將中斷序列化的陳述式，其會遭到停止並進行復原。每次發生交易衝突，Amazon Redshift 就會將資料列寫入至 STL_TR_CONFLICT 系統資料表，其中包含已取消之交易的詳細資訊。如需詳細資訊，請參閱 可序列化隔離。

只有超級使用者才能看到 STL_TR_CONFLICT。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_TRANSACTION_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
xact_id	bigint	已復原之交易的交易 ID。
process_id	bigint	與已復原之交易相關聯的處理程序。
xact_start_ts	timestamp	交易開始時的時間戳記 (UTC)。
abort_time	timestamp	交易停止時的時間戳記 (UTC)。
table_id	bigint	發生衝突之資料表的資料表 ID。

範例查詢

若要傳回涉及特定資料表之衝突的相關資訊，請執行一個指定資料表 ID 的查詢：

```
select * from stl_tr_conflict where table_id=100234
```

```
order by xact_start_ts;

xact_id|process_|      xact_start_ts       |        abort_time        |table_
       |id      |                          |                          |id
-------+--------+--------------------------+--------------------------+------
  1876 |  8551  |2010-03-30 09:19:15.852326|2010-03-30 09:20:17.582499|100234
  1928 | 15034  |2010-03-30 13:20:00.636045|2010-03-30 13:20:47.766817|100234
  1991 | 23753  |2010-04-01 13:05:01.220059|2010-04-01 13:06:06.94098 |100234
  2002 | 23679  |2010-04-01 13:17:05.173473|2010-04-01 13:18:27.898655|100234
(4 rows)
```

您可以從序列化違規之錯誤訊息的 DETAIL 區段取得資料表 ID (錯誤 1023)。

STL_UNDONE

顯示已復原之交易的相關資訊。

所有使用者都可看見 STL_UNDONE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_TRANSACTION_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xact_id	bigint	復原交易的 ID。
xact_id_undone	bigint	已復原之交易的 ID。
undo_start_ts	timestamp	復原交易的開始時間。
undo_end_ts	timestamp	復原交易的結束時間。
table_id	bigint	受到復原交易影響之資料表的 ID。

範例查詢

若要檢視所有已復原之交易的精簡日誌，請輸入下列命令：

```
select xact_id, xact_id_undone, table_id from stl_undone;
```

此命令會傳回下列範例輸出：

```
 xact_id | xact_id_undone | table_id
---------+----------------+----------
1344 |           1344 |   100192
1326 |           1326 |   100192
1551 |           1551 |   100192
(3 rows)
```

STL_UNIQUE

分析在 SELECT 清單中使用 DISTINCT 函數時發生的執行步驟，或在 UNION 或 INTERSECT 查詢中複製或移除時發生的執行步驟。

所有使用者都可看見 STL_UNIQUE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_UNIQUE 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。

欄名稱	資料類型	描述
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
type	character(6)	步驟的類型。有效的 值如下： • HASHED。表示步驟已使用分組且未排序的彙整。 • PLAIN。表示步驟已使用未分組的純量彙整。 • SORTED。表示步驟已使用分組且排序的彙整。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
slots	integer	雜湊儲存貯體的總數。
workmem	bigint	已指派給步驟之運作中記憶體中的位元組總數。
max_buffers_used	bigint	在移至磁碟之前於雜湊表中使用的緩衝區數目上限。
resizes	integer	此資訊僅供內部使用。
occupied	integer	此資訊僅供內部使用。
flushable	integer	此資訊僅供內部使用。

欄名稱	資料類型	描述
used_unique_prefetching	character(1)	此資訊僅供內部使用。
位元組	biginit	步驟之所有輸出列的位元組數目。

範例查詢

假設您執行下列查詢：

```
select distinct eventname
from event order by 1;
```

假設前一個查詢的 ID 為 6313，下列範例會顯示唯一步驟針對區段 0 與 1 中之每個配量所產生的資料列數目。

```
select query, slice, segment, step, datediff(msec, starttime, endtime) as msec,
 tasknum, rows
from stl_unique where query = 6313
order by query desc, slice, segment, step;
```

```
 query | slice | segment | step | msec | tasknum | rows
-------+-------+---------+------+------+---------+------
  6313 |     0 |       0 |    2 |    0 |      22 |  550
  6313 |     0 |       1 |    1 |  256 |      20 |  145
  6313 |     1 |       0 |    2 |    1 |      23 |  540
  6313 |     1 |       1 |    1 |   42 |      21 |  127
  6313 |     2 |       0 |    2 |    1 |      22 |  540
  6313 |     2 |       1 |    1 |  255 |      20 |  158
  6313 |     3 |       0 |    2 |    1 |      23 |  542
  6313 |     3 |       1 |    1 |   38 |      21 |  146
(8 rows)
```

STL_UNLOAD_LOG

記錄卸載操作的詳細資訊。

STL_UNLOAD_LOG 會針對 UNLOAD 陳述式所建立的每一個檔案記錄一個資料列。例如，若 UNLOAD 建立 12 個檔案，則 STL_UNLOAD_LOG 將包含 12 個對應資料列。

所有使用者都可看見 STL_UNLOAD_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_UNLOAD_LOG 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_UNLOAD_HISTORY 和 SYS_UNLOAD_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。
分割	integer	識別執行查詢之配量的數字。
pid	integer	與查詢陳述式相關聯的處理程序 ID。
路徑	character(1280)	檔案的完整 Amazon S3 物件路徑。
start_time	timestamp	交易的開始時間。
end_time	timestamp	交易的結束時間。
line_count	bigint	已卸載至檔案的行數 (列數)。
transfer_size	bigint	已傳輸的位元組數目。
file_format	character(10)	已卸載檔案的格式。

範例查詢

若要取得透過 UNLOAD 命令寫入 Amazon S3 的檔案清單，您可以在 UNLOAD 完成後呼叫 Amazon S3 清單操作。您也可以查詢 STL_UNLOAD_LOG。

下列查詢會傳回已由 UNLOAD 針對前一個完成的查詢建立之檔案的路徑名稱：

```
select query, substring(path,0,40) as path
from stl_unload_log
where query = pg_last_query_id()
order by path;
```

此命令會傳回下列範例輸出：

```
 query |             path
-------+---------------------------------------
  2320 | s3://my-bucket/venue0000_part_00
  2320 | s3://my-bucket/venue0001_part_00
  2320 | s3://my-bucket/venue0002_part_00
  2320 | s3://my-bucket/venue0003_part_00
(4 rows)
```

STL_USAGE_CONTROL

STL_USAGE_CONTROL 檢視包含達到用量限制時所記錄的資訊。如需使用限制的相關資訊，請參閱《Amazon Redshift 管理指南》中的管理用量限制。

只有超級使用者才能看到 STL_USAGE_CONTROL。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
eventtime	timestamp	查詢超過用量限制的時間 (UTC)。
query	integer	查詢識別符。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
xid	bigint	交易識別符。

欄名稱	資料類型	描述
pid	integer	與查詢相關聯的處理程序識別碼。
usage_limit_id	character(40)	Amazon Redshift 產生的全域唯一識別碼 (UUID)，例如 25d9297e-3e7b-41c8-9f4d-c4b6eb731c09 。
feature_type	character(30)	超過用量限制的功能。可能的值包括 CONCURRENCY_SCALING 和 SPECTRUM。

範例查詢

下列 SQL 範例會傳回一些達到用量限制時記錄的資訊。

```
select query, pid, eventtime, feature_type
from stl_usage_control
order by eventtime desc
limit 5;
```

STL_USERLOG

記錄資料庫使用者之下列變更的詳細資訊：

- 建立使用者
- 捨棄使用者
- 更改使用者 (重新命名)
- 更改使用者 (更改屬性)

只有超級使用者才能看到 STL_USERLOG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_USERLOG 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	受到變更影響之使用者的 ID。
使用者名稱	character(50)	受到變更影響之使用者的使用者名稱。
oldusername	character(50)	若為重新命名動作，這是原始使用者名稱。若是任何其他動作，此欄位空白。
動作	character(10)	發生的動作。有效值： • Alter • 建立 • Drop • 重新命名
usecreatedb	integer	若為 true (1)，表示使用者具備建立資料庫權限。
usesuper	integer	若為 true (1)，表示使用者是超級使用者。
usecatupd	integer	若為 true (1)，表示使用者可以更新系統目錄。
valuntil	timestamp	密碼到期日。
pid	integer	處理程序 ID。
xid	bigint	交易 ID。
recordtime	timestamp	查詢開始的時間，以 UTC 表示。

範例查詢

下列範例會執行四個使用者動作，然後查詢 STL_USERLOG 檢視。

```
create user userlog1 password 'Userlog1';
alter user userlog1 createdb createuser;
```

```
alter user userlog1 rename  to userlog2;
drop user userlog2;

select userid, username, oldusername, action, usecreatedb, usesuper from stl_userlog
 order by recordtime desc;
```

```
 userid |  username | oldusername |  action | usecreatedb | usesuper
--------+-----------+-------------+---------+-------------+----------
    108 | userlog2  |             | drop    |           1 |   1
    108 | userlog2  | userlog1    | rename  |           1 |   1
    108 | userlog1  |             | alter   |           1 |   1
    108 | userlog1  |             | create  |           0 |   0
 (4 rows)
```

STL_UTILITYTEXT

擷取在資料庫上執行之非 SELECT SQL 命令的文字。

查詢 STL_UTILITYTEXT 檢視，以擷取已在系統上執行之 SQL 陳述式的下列子集：

- ABORT、BEGIN、COMMIT、END、ROLLBACK
- ANALYZE
- CALL
- 取消
- COMMENT
- CREATE、ALTER、DROP DATABASE
- CREATE、ALTER、DROP USER
- EXPLAIN
- GRANT、REVOKE
- LOCK
- RESET
- SET
- SHOW
- TRUNCATE

另請參閱 STL_DDLTEXT、STL_QUERYTEXT 和 SVL_STATEMENTTEXT。

使用 STARTTIME 和 ENDTIME 資料欄，來了解已在特定時段記錄哪些陳述式。SQL 文字的長區塊會分成數行，一行 200 個字元；SEQUENCE 欄會識別屬於單一陳述式的文字片段。

所有使用者都可看見 STL_UTILITYTEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	交易 ID。
pid	integer	與查詢陳述式相關聯的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位為空白。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回已在 2012 年 1 月 26 日執行之「公用程式」命令的文字。在此案例中，系統已執行部分 SET 命令和 SHOW ALL 命令：

```
select starttime, sequence, rtrim(text)
from stl_utilitytext
where starttime like '2012-01-26%'
order by starttime, sequence;

starttime          | sequence |              rtrim
---------------------------+-----+---------------------------------
2012-01-26 13:05:52.529235 |   0 | show all;
2012-01-26 13:20:31.660255 |   0 | SET query_group to ''
2012-01-26 13:20:54.956131 |   0 | SET query_group to 'soldunsold.sql'
...
```

重建儲存的 SQL

若要重建儲存在 STL_UTILITYTEXT `text` 資料欄中的 SQL，則需執行 SELECT 陳述式以從 `text` 資料欄的一或多個部分建立 SQL。請先以新的一行取代任意 (\n) 特殊字元，再執行重建的 SQL。下列 SELECT 陳述式顯示的結果會是 `query_statement` 欄中重建的 SQL 資料列。

```
SELECT LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END) WITHIN
 GROUP (ORDER BY sequence) as query_statement
FROM stl_utilitytext GROUP BY xid order by xid;
```

例如，下列查詢會將 query_group 設為一個零的字串。該查詢本身的長度超過 200 個字元，且會儲存在 STL_UTILITYTEXT 的多個部分中。

```
set query_group to
 '0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
             000000';
```

在本範例中，查詢會儲存在 STL_UTILITYTEXT `text` 資料欄的 2 個部分 (資料列) 中。

```
select query, sequence, text
from stl_utilitytext where query=pg_last_query_id() order by query desc, sequence limit
 10;
```

```
         starttime          | sequence |
                                                text

----------------------------+----------
+-------------------------------------------------------------------------------------------------------
 2019-07-23 22:55:34.926198 |        0 | set query_group to
 '000000000000000000000000000000000000000000000000000000000000000000000000000000000000\n00000000000
 2019-07-23 22:55:34.926198 |        1 |                   000000';
```

若要重建儲存在 STL_UTILITYTEXT 中的 SQL，請執行下列 SQL。

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS query_statement
from stl_utilitytext where query=pg_last_query_id();
```

若要在用戶端中使用產生的重建 SQL，請以新的一行取代任意 (\n) 特殊字元。

```
                                              query_statement


-------------------------------------------------------------------------------------------------------
 set query_group to
 '000000000000000000000000000000000000000000000000000000000000000000000000000000000000\n00000000000
                  000000';
```

STL_VACUUM

顯示已清空之資料表的資料列和區塊統計資訊。

此檢視顯示每個清空操作啟動和完成時特有的資訊，並示範執行此操作的優點。如需執行此命令之需求的相關資訊，請參閱 VACUUM 命令說明。

只有超級使用者才能看到 STL_VACUUM。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_VACUUM_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	VACUUM 陳述式的交易 ID。您可以將此檢視聯結至 STL_QUERY 檢視，以查看針對特定 VACUUM 交易執行的個別 SQL 陳述式。如果清空整個資料庫，則會清空個別交易中的每個資料表。
table_id	integer	資料表 ID。
status	character(30)	每個資料表之 VACUUM 操作的狀態。可能的值如下： • **Started** • **Started Delete Only** • **Started Delete Only (Sorted >= nn%)** 僅對 VACUUM FULL 啟動刪除階段。已略過排序階段，因為排序的資料表已達或超過排序閾值。 • **Started Sort Only** • **Started Ranged Partition** • **Started Reindex** • **Finished** 已對資料表完成操作的時間。若要了解清空作業已對特定資料表花費了多長時間，請從特定交易 ID 和資料表 ID 的完成時間中減去啟動時間。 • **Skipped** 已略過資料表，因為已完全排序資料表，而且未標示任何資料列進行刪除。 • **Skipped (delete only)** 已略過資料表，因為已指定 DELETE ONLY，而且未標示任何資料列進行刪除。

欄名稱	資料類型	描述
		• **Skipped (sort only)** 已略過資料表，因為已指定 SORT ONLY，而且已完全排序資料表。 • **Skipped (sort only, sorted>=xx%)** 已略過資料表，因為已指定 SORT ONLY，而且排序的資料表已達或超過排序閾值。 • **Skipped (0 rows)** 已略過資料表，因為它是空的。 • **VacuumBG** 自動清空操作會在背景執行。當其他狀態自動執行時，此狀態會附加在其他狀態之前。例如，自動執行的僅刪除清空將具有狀態 `[VacuumBG] Started Delete Only` 的起始列。 如需 VACUUM 排序閾值設定的相關資訊，請參閱 VACUUM。
rows	bigint	資料表中的實際資料列數目加上任何仍在磁碟上儲存的已刪除資料列 (等待清空)。此欄顯示在對狀態為 **Started** 的資料列啟動清空之前的計數，以及在對狀態為 **Finished** 的資料列啟動清空之後的計數。
sortedrows	integer	資料表中已排序的資料列數目。此欄顯示在對 Status 欄中具有 **Started** 的資料列啟動清空之前的計數，以及在對 Status 欄中具有 **Finished** 的資料列啟動清空之後的計數。
blocks	integer	在清空操作 (狀態為 **Started** 的資料列) 之前，以及在清空操作 (**Finished** 欄) 之後，用來儲存資料表資料的資料區塊總數。每個資料區塊都使用 1 MB。

欄名稱	資料類型	描述
max_merge_partitions	integer	會使用此欄位來分析表現並表示分割區數上限，也就是清空可以根據合併階段反覆運算為資料表處理的分割區數上限。(清空會將未排序的區域排序為一個或多個已排序的分割區。根據資料表中的欄數和目前的 Amazon Redshift 組態，合併階段可以在單次合併迭代中處理最大數量的分割區。如果排序分割區的數量超過合併分割區的最大數量，合併階段仍然有效，但需要更多的合併迭代。)
eventtime	timestamp	清空操作啟動或完成的時間。
reclaimable_rows	bigint	目前 cutoff_xid 的可回收列數。此欄顯示 Redshift 在清空開始之前對於具有 **Started** 狀態之列的估計可回收列數，以及在清空之後對於具有 **Finished** 狀態之列剩餘的實際可回收列數。
reclaimable_space_mb	bigint	目前 cutoff_xid 的可回收空間 (以 MB 為單位)。此欄顯示 Redshift 在清空開始之前對於具有 **Started** 狀態之列的估計可回收空間量，以及在清空之後對於具有 **Finished** 狀態之列剩餘的實際可回收空間量。
cutoff_xid	bigint	VACUUM 操作的截止交易 ID。截止後的任何交易都不包括在 VACUUM 操作中。
is_recluster	integer	如果為 1 (true)，則 VACUUM 操作執行重新建立叢集演算法，如果為 0 (false)，則不執行。

範例查詢

下列查詢報告資料表 108313 的清空統計資訊。在一系列插入和刪除之後已清空此資料表。

```
select xid, table_id, status, rows, sortedrows, blocks, eventtime,
                reclaimable_rows, reclaimable_space_mb
from stl_vacuum where table_id=108313 order by eventtime;

 xid   | table_id | status                    | rows | sortedrows | blocks |  eventtime
         | reclaimable_rows | reclaimable_space_mb
```

```
-------+----------+-------------------------+------+-----------+--------
+---------------------+-----------------+----------------------
 14294 |   108313 | Started                 | 1950 |       408 |     28 |  2016-05-19
 17:36:01 |              984 |                   17
 14294 |   108313 | Finished                |  966 |       966 |     11 |  2016-05-19
 18:26:13 |                0 |                    0
 15126 |   108313 | Skipped(sorted>=95%)    |  966 |       966 |     11 |  2016-05-19
 18:26:38 |                0 |                    0
```

在 VACUUM 開始時，資料表包含 1,950 列，儲存在 28 個 1 MB 區塊中。Amazon Redshift 估計，透過清空操作可以回收 984 個磁碟空間，即 17 個磁碟空間區塊。

在「已完成」狀態的列中，ROWS 欄顯示的值為 966，而 BLOCKS 欄的值從 28 下降為 11。清空回收了估計的磁碟空間量，清空操作完成後沒有剩餘可回收的列或空間。

在排序階段 (交易 15126) 中，清空能夠略過資料表，因為已依排序索引鍵順序插入資料列。

下列範例會在大型 INSERT 操作之後顯示 SALES 資料表 (此範例中的資料表 110116) 上 SORT ONLY 清空的統計資訊：

```
vacuum sort only sales;

select xid, table_id, status, rows, sortedrows, blocks, eventtime
from stl_vacuum order by xid, table_id, eventtime;

xid |table_id|     status      | rows  |sortedrows|blocks|     eventtime
----+--------+-----------------+-------+----------+------+--------------------
...
2925| 110116 |Started Sort Only|1379648|   172456 |  132 | 2011-02-24 16:25:21...
2925| 110116 |Finished         |1379648|  1379648 |  132 | 2011-02-24 16:26:28...
```

STL_WINDOW

分析執行視窗函數的查詢步驟。

所有使用者都可看見 STL_WINDOW。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> STL_WINDOW 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
endtime	timestamp	查詢完成的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體中的位元組總數。

範例查詢

下列範例傳回配量 0 和區段 3 的視窗函數結果。

```
select query, tasknum, rows, is_diskbased, workmem
from stl_window
where slice=0 and segment=3;
```

```
 query | tasknum | rows | is_diskbased | workmem
-------+---------+------+--------------+----------
 86326 |      36 | 1857 | f            | 95256616
   705 |      15 | 1857 | f            | 95256616
 86399 |      27 | 1857 | f            | 95256616
   649 |      10 |    0 | f            | 95256616
(4 rows)
```

STL_WLM_ERROR

記錄所有發生的 WLM 相關錯誤。

所有使用者都可看見 STL_WLM_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
recordtime	timestamp	發生錯誤的時間。
pid	integer	產生錯誤之處理程序的 ID。
error_string	character(256)	錯誤說明。

STL_WLM_RULE_ACTION

記錄 WLM 查詢監控規則所產生之動作的詳細資訊，而此規則與使用者定義的查詢相關聯。如需詳細資訊，請參閱 WLM 查詢監控規則。

所有使用者都可看見 STL_WLM_RULE_ACTION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行查詢的使用者。
query	integer	查詢 ID。
service_class	integer	服務類別的 ID。查詢佇列定義於 WLM 組態。大於 5 的服務類別是使用者定義的佇列。
rule	character(256)	查詢監控規則的名稱。
動作	character(256)	產生的動作。可能的值如下： • log • hop(reassign) • hop(restart) • abort • change_query_priority • 無 none 值表示已符合規則的述詞，但此動作已被另一個規則取代為嚴重性更高的動作。
recordtime	timestamp	動作以 UTC 記錄的時間。
action_value	character(256)	若 action 為 change_query_priority ，則可能的值會是 highest、high、normal、low 和 lowest。 若 action 為 log、hop 或 abort，則此值為是空白。
service_class_name	character(64)	服務類別的名稱。

範例查詢

下列範例尋找查詢監控規則已停止的查詢。

```
Select query, rule
from stl_wlm_rule_action
where action = 'abort'
order by query;
```

STL_WLM_QUERY

在 WLM 處理的服務類別中包含每個嘗試執行之查詢的記錄。

所有使用者都可看見 STL_WLM_QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	integer	查詢或子查詢的交易 ID。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可與多個查詢 ID 關聯。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
service_class	integer	服務類別的 ID。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
slot_count	integer	根據為佇列設定的並行層級，查詢使用的 WLM 查詢槽數。預設值為 1。如需詳細資訊，請參閱 wlm_query_slot_count。

欄名稱	資料類型	描述
service_class_start_time	timestamp	查詢已指派給服務類別的時間。此時間位於 UTC 時區。
queue_start_time	timestamp	查詢進入服務類別佇列的時間。此時間位於 UTC 時區。
queue_end_time	timestamp	查詢離開服務類別佇列的時間。此時間位於 UTC 時區。
total_queue_time	bigint	查詢在佇列中花費的微秒總數
exec_start_time	timestamp	查詢開始在服務類別執行的時間。此時間位於 UTC 時區。
exec_end_time	timestamp	查詢在服務類別完成執行的時間。此時間位於 UTC 時區。
total_exec_time	bigint	查詢花費在執行的微秒數。
service_class_end_time	timestamp	查詢離開服務類別的時間。此時間位於 UTC 時區。
final_state	character(16)	保留以供系統使用。
est_peak_mem	bigint	保留以供系統使用。
query_priority	char(20)	查詢的優先順序。可能的值為 n/a、lowest、low、normal、high 和 highest，其中 n/a 表示不支援查詢優先順序。
service_class_name	character(64)	服務類別名稱。如需服務類別的相關資訊，請參閱 WLM 系統資料表和檢視。

範例查詢

檢視佇列和執行中的平均查詢時間

下列查詢顯示大於 4 之服務類別的目前組態。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。

下列查詢會傳回每個查詢花費在查詢佇列以及執行每個服務類別的平均時間 (以微秒為單位)。

```
select service_class as svc_class, count(*),
avg(datediff(microseconds, queue_start_time, queue_end_time)) as avg_queue_time,
avg(datediff(microseconds, exec_start_time, exec_end_time )) as avg_exec_time
from stl_wlm_query
where service_class > 4
group by service_class
order by service_class;
```

此查詢傳回下列範例輸出：

```
 svc_class | count | avg_queue_time | avg_exec_time
-----------+-------+----------------+---------------
         5 | 20103 |              0 |         80415
         5 |  3421 |          34015 |        234015
         6 |    42 |              0 |        944266
         7 |   196 |           6439 |       1364399
(4 rows)
```

檢視佇列和執行中的查詢時間上限

下列查詢會傳回每個查詢花費在任何查詢佇列以及執行每個服務類別的時間量上限 (以微秒為單位)。

```
select service_class as svc_class, count(*),
max(datediff(microseconds, queue_start_time, queue_end_time)) as max_queue_time,
max(datediff(microseconds, exec_start_time, exec_end_time )) as max_exec_time
from stl_wlm_query
where svc_class > 5
group by service_class
order by service_class;
```

```
 svc_class | count | max_queue_time | max_exec_time
-----------+-------+----------------+---------------
         6 |    42 |              0 |       3775896
         7 |   197 |          37947 |      16379473
(4 rows)
```

快照資料的 STV 資料表

STV 資料表是虛擬系統資料表，包含目前系統資料的快照。

主題

- STV_ACTIVE_CURSORS
- STV_BLOCKLIST
- STV_CURSOR_CONFIGURATION
- STV_DB_ISOLATION_LEVEL
- STV_EXEC_STATE
- STV_INFLIGHT
- STV_LOAD_STATE
- STV_LOCKS
- STV_ML_MODEL_INFO
- STV_MV_DEPS
- STV_MV_INFO
- STV_NODE_STORAGE_CAPACITY
- STV_PARTITIONS
- STV_QUERY_METRICS
- STV_RECENTS
- STV_SESSIONS
- STV_SLICES
- STV_STARTUP_RECOVERY_STATE
- STV_TBL_PERM
- STV_TBL_TRANS
- STV_WLM_CLASSIFICATION_CONFIG
- STV_WLM_QMR_CONFIG
- STV_WLM_QUERY_QUEUE_STATE
- STV_WLM_QUERY_STATE
- STV_WLM_QUERY_TASK_STATE
- STV_WLM_SERVICE_CLASS_CONFIG
- STV_WLM_SERVICE_CLASS_STATE

- STV_X 還原 _ 交換佇列狀態

STV_ACTIVE_CURSORS

STV_ACTIVE_CURSORS 顯示目前開啟之游標的詳細資訊。如需詳細資訊，請參閱 DECLARE。

所有使用者都可看見 STV_ACTIVE_CURSORS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。使用者只能檢視該使用者所開啟的游標。超級使用者可檢視所有游標。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
name	characte (256)	游標名稱。
xid	bigint	交易細節。
pid	integer	執行查詢的前置程序。
starttime	timestam	宣告游標的時間。
row_count	bigint	游標結果集中的資料列數。
byte_count	bigint	游標結果集中的位元組數。
fetched_rows	bigint	目前從游標結果集擷取的資料列數。

STV_BLOCKLIST

STV_BLOCKLIST 包含資料庫中各個分割、資料表或欄位所使用的 1 MB 磁碟區塊數。

使用彙總查詢與 STV_BLOCKLIST (如下範例所示) 以判斷每個資料庫、資料表、分割或欄位所配置的 1 MB 磁碟區塊數。您也可以使用 STV_PARTITIONS 來檢視磁碟使用率的摘要資訊。

只有超級使用者可以看到 STV_BLOCKLIST。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
分割	integer	節點分割。
col	integer	欄位的以零為起始的索引。您建立的每個資料表皆附加三個隱藏欄位：INSERT_XID、DELETE_XID 及 ROW_ID (OID)。包含 3 個使用者定義欄位的資料表包含 6 個實際欄位，使用者定義的欄位的內部編號為 0、1 及 2。在此範例中，INSERT_XID、DELETE_XID 及 ROW_ID 欄位的編號分別為 3、4 及 5。
tbl	integer	資料庫資料表的資料表 ID。
blocknum	integer	資料區塊的 ID。
num_values	integer	區塊所包含的值的數量。
extended_limits	integer	供內部使用。
minvalue	bigint	區塊的最小資料值。儲存前 8 個字元做為非數值資料的 64 位元整數。用於磁碟掃描。
maxvalue	bigint	區塊的最大資料值。儲存前 8 個字元做為非數值資料的 64 位元整數。用於磁碟掃描。
sb_pos	integer	磁碟上的超級區塊位置的內部 Amazon Redshift 識別碼。
pinned	integer	區塊是否固定至記憶體做為預載的一部分。0 = false；1 = true。預設為 false。
on_disk	integer	區塊是否已自動儲存於磁碟。0 = false；1 = true。預設為 false。
modified	integer	區塊是否已修改。0 = false；1 = true。預設為 false。

欄名稱	資料類型	描述
hdr_modified	integer	區塊標頭是否已修改。0 = false；1 = true。預設為 false。
unsorted	integer	區塊是否未排序。0 = false；1 = true。預設為 true。
tombstone	integer	供內部使用。
preferred_diskno	integer	區塊應處於開啟狀態的磁碟數量 (無論磁碟是否故障)。一旦磁碟修復，區塊將移回該磁碟。
temporary	integer	無論區塊是否包含暫存資料，例如來自暫存資料表或中繼查詢結果。0 = false；1 = true。預設為 false。
newblock	integer	指出區塊是否是新的 (true) 或不曾遞交至磁碟 (false)。0 = false；1 = true。
num_readers	integer	每個區塊上的參考數量。
flags	integer	區塊標頭檔內部 Amazon Redshift 旗標。

範例查詢

STV_BLOCKLIST 在每個配置的磁碟區塊上包含一個資料列，因此選取所有資料列的查詢可能會傳回數量非常大的資料列。建議僅適用彙總查詢搭配 STV_BLOCKLIST。

SVV_DISKUSAGE 檢視以更友善使用者的格式提供類似的資訊；但是，以下範例示範 STV_BLOCKLIST 資料表的其中一種用法。

若要判斷 VENUE 資料表中各個欄位使用的 1 MB 區塊數，請輸入以下查詢：

```
select col, count(*)
from stv_blocklist, stv_tbl_perm
where stv_blocklist.tbl = stv_tbl_perm.id
and stv_blocklist.slice = stv_tbl_perm.slice
and stv_tbl_perm.name = 'venue'
group by col
```

```
order by col;
```

此查詢會傳回配置於 VENUE 資料表中各個欄位的 1 MB 磁碟區塊數，以下列範例資料呈現：

```
 col | count
-----+-------
   0 |  4
   1 |  4
   2 |  4
   3 |  4
   4 |  4
   5 |  4
   7 |  4
   8 |  4
(8 rows)
```

以下查詢顯示資料表資料是否實際發佈至所有分割：

```
select trim(name) as table, stv_blocklist.slice, stv_tbl_perm.rows
from stv_blocklist,stv_tbl_perm
where stv_blocklist.tbl=stv_tbl_perm.id
and stv_tbl_perm.slice=stv_blocklist.slice
and stv_blocklist.id > 10000 and name not like '%#m%'
and name not like 'systable%'
group by name, stv_blocklist.slice, stv_tbl_perm.rows
order by 3 desc;
```

此查詢會產生下列範例輸出，顯示具有最多資料列之資料表的偶數資料分佈：

```
table   | slice | rows
----------+-------+-------
listing  |    13 | 10527
listing  |    14 | 10526
listing  |     8 | 10526
listing  |     9 | 10526
listing  |     7 | 10525
listing  |     4 | 10525
listing  |    17 | 10525
listing  |    11 | 10525
listing  |     5 | 10525
listing  |    18 | 10525
```

```
listing  |    12 | 10525
listing  |     3 | 10525
listing  |    10 | 10525
listing  |     2 | 10524
listing  |    15 | 10524
listing  |    16 | 10524
listing  |     6 | 10524
listing  |    19 | 10524
listing  |     1 | 10523
listing  |     0 | 10521
...
(180 rows)
```

下列查詢可判斷是否有任何已刪除標記的區塊已遞交至磁碟：

```
select slice, col, tbl, blocknum, newblock
from stv_blocklist
where  tombstone > 0;

slice | col |   tbl  | blocknum | newblock
-------+-----+--------+----------+----------
4     |  0  | 101285 |    0     |   1
4     |  2  | 101285 |    0     |   1
4     |  4  | 101285 |    1     |   1
5     |  2  | 101285 |    0     |   1
5     |  0  | 101285 |    0     |   1
5     |  1  | 101285 |    0     |   1
5     |  4  | 101285 |    1     |   1
...
(24 rows)
```

STV_CURSOR_CONFIGURATION

STV_CURSOR_CONFIGURATION 顯示游標組態限制。如需詳細資訊，請參閱 游標限制條件。

只有超級使用者可以看到 STV_CURSOR_CONFIGURATION。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
current_cursor_count	integer	目前已開啟的游標數量。
max_diskspace_usable	integer	可供游標使用之磁碟空間量，以 MB 為單位。此限制根據叢集的最大游標結果集大小而定。
current_diskspace_used	integer	目前游標使用之磁碟空間量，以 MB 為單位。

STV_DB_ISOLATION_LEVEL

STV_DB_ISOLATION_LEVEL 會顯示資料庫目前的隔離層級。如需隔離層級的相關資訊，請參閱CREATE DATABASE。

所有使用者都可看見 STV_DB_ISOLATION_LEVEL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
db_name	character(128)	資料庫名稱。
isolation_level	character(20)	資料庫的隔離層級。可能的值包括 `Serializable` 和 `Snapshot Isolation` 。

STV_EXEC_STATE

使用 STV_EXEC_STATE 資料表以找出有關正在運算節點執行之查詢及查詢步驟的資訊。

此資訊通常僅用於排解工程設計問題。SVV_QUERY_STATE 與 SVL_QUERY_SUMMARY 檢視會從 STV_EXEC_STATE 擷取其資訊。

所有使用者都可看見 STV_EXEC_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
分割	integer	步驟完成的節點分割。
segment	integer	執行查詢的區段。查詢區段是一系列的步驟。
step	integer	完成查詢區段的步驟。步驟是查詢執行的最小單位。
starttime	timestamp	步驟執行的時間。
currenttime	timestamp	目前時間。
tasknum	integer	指派完成步驟之查詢任務程序。
rows	bigint	處理的資料列數。
位元組	bigint	處理的位元組數。
label	char(256)	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 `scan tbl=100448 name =user`)。三位數資料表 ID 通常是指暫時性資料表的掃描。當您看見 `tbl=0` 時，通常是指常數值的掃描。
is_diskbased	char(1)	查詢的這個步驟是否以磁碟型操作方式完成：true (**t**) 或 false (**f**)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中完成。

欄名稱	資料類型	描述
workmem	bigint	已指派給步驟之運作中記憶體的位元組數。
num_parts	integer	雜湊資料表在雜湊步驟期間已分割的分割區數。此欄位中的正數並不表示雜湊步驟以磁碟式操作方式執行。檢查 IS_DISKBASED 欄位中的值，查看該雜湊步驟是否為磁碟型。
is_rrscan	char(1)	若為 true (**t**)，表示已在步驟上使用範圍限制掃描。預設為 false (**f**)。
is_delayed_scan	char(1)	若為 true (**t**)，表示已在該步驟上使用延遲的掃描。預設為 false (**f**)。

範例查詢

Amazon Redshift 建議查詢 SVL_QUERY_SUMMARY 或 SVV_QUERY_STATE 以更友善使用者的格式獲得 STV_EXEC_STATE 中的資訊，而非直接查詢 STV_EXEC_STATE。如需詳細資訊，請參閱 SVL_QUERY_SUMMARY 或 SVV_QUERY_STATE 資料表文件。

STV_INFLIGHT

使用 STV_INFLIGHT 資料表來判斷正在叢集中執行的查詢為何。如果您要進行疑難排解，檢查長時間執行查詢的狀態會很有幫助。

STV_INFLIGHT 不會顯示僅限於領導節點的查詢。如需詳細資訊，請參閱 僅限領導節點函數。所有使用者都可看見 STV_INFLIGHT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

使用 STV_INFLIGHT 進行疑難排解

如果您使用 STV_INFLIGHT 對查詢或查詢集合的效能進行疑難排解，請注意下列事項：

- 長時間運行的開放交易通常會增加負載。這些開啟的交易可能會導致其他查詢的執行時間較長。
- 長時間執行的 COPY 和 ETL 任務佔用大量運算資源，可能會影響在叢集上執行的其他查詢。在大多數情況下，將這些長時間執行的工作移至低使用率的時間，可提高報告或分析工作負載的效能。

- 有些檢視會提供相關資訊給 STV_INFLIGHT。其中包括擷取 SQL 命令查詢文字的 STL_QUERYTEXT，以及將 STV_INFLIGHT 連接至 STL_QUERYTEXT 的 SVV_QUERY_INFLIGHT。您也可以搭配 STV_RECENTS 使用 STV_INFLIGHT 進行疑難排解。例如，STV_RECENTS 可以指出特定查詢處於執行中或完成狀態。將這些資訊與 STV_INFLIGHT 的結果結合在一起，可以為您提供有關查詢屬性和運算資源影響的更多資訊。

您也可以使用 Amazon Redshift 主控台監控執行中的查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
分割	integer	查詢執行時所在的分割。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位為空白。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。您可以使用此資料欄來聯結至 STL_ERROR 資料表。
starttime	timestamp	查詢開始的時間。
text	character(100)	查詢文字，如果陳述式超過限制，將會截斷為 100 個字元。
suspended	integer	查詢是否已遭停用。0 = false；1 = true。
insert_pristine	integer	當目前查詢正在執行時，寫入查詢是否能夠執行。1 = 不允許寫入查詢。0 = 允許寫入查詢。此欄位適用於偵錯。

欄名稱	資料類型	描述
concurrency_scaling_status	integer	指出查詢是執行於主要叢集或並行擴展叢集，可能的值如下： 0 - 執行於主要叢集 1 - 執行於並行擴展叢集

範例查詢

若要檢視目前正在資料庫上執行的所有使用中查詢，請輸入下列查詢：

```
select * from stv_inflight;
```

以下範例輸出顯示有兩個查詢正在執行，包括 STV_INFLIGHT 查詢本身以及從名為 avgwait.sql 的指令碼執行的查詢：

```
select slice, query, trim(label) querylabel, pid,
starttime, substring(text,1,20) querytext
from stv_inflight;

slice|query|querylabel | pid |        starttime         |      querytext
-----+-----+-----------+-----+--------------------------+--------------------
1011 |  21 |           | 646 |2012-01-26 13:23:15.645503|select slice, query,
1011 |  20 |avgwait.sql| 499 |2012-01-26 13:23:14.159912|select avg(datediff(
(2 rows)
```

下列查詢會選擇幾個欄，包括 concurrency_scaling_status。此欄指出是否要將查詢傳送至並行擴展叢集。如果該值對於某些結果是 1，則表示正在使用並行擴展運算資源。如需詳細資訊，請參閱 使用並行擴展。

```
select userid,
query,
pid,
starttime,
text,
suspended,
concurrency_scaling_status
 from STV_INFLIGHT;
```

範例輸出會顯示傳送至並行擴展叢集的一個查詢。

```
 query  | pid     |        starttime           |   text                  | suspended
 |  concurrency_scaling_status
--------+---------
+---------------------------|-----------------------|--------------|-----------------------
1234567 | 123456  | 2012-01-26 13:23:15.645503 | select userid, query...  0
    1
2345678 | 234567  | 2012-01-26 13:23:14.159912 | select avg(datediff(...  0
    0
(2 rows)
```

如需疑難排解查詢效能的詳細提示，請參閱對查詢進行故障診斷。

STV_LOAD_STATE

使用 STV_LOAD_STATE 資料表以尋找有關進行中的 COPY 陳述式目前狀態的資訊。

COPY 命令會在每載入一百萬筆記錄之後更新此資料表。

所有使用者都可看見 STV_LOAD_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
session	integer	執行此載入處理程序的工作階段 PID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
分割	integer	節點分割數量。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。
recordtime	timestamp	記錄被記錄的時間。

欄名稱	資料類型	描述
bytes_to_load	bigint	此分割要載入的位元組總數。如果要載入的資料已壓縮，此數值為 0
bytes_loaded	bigint	此分割要載入的位元組數。如果要載入的資料已壓縮，此數值為資料解壓縮之後載入的位元組數。
bytes_to_load_compressed	bigint	此分割要載入的已壓縮資料的位元組總數。如果要載入的資料未壓縮，此數值為 0。
bytes_loaded_compressed	bigint	此分割要載入的已壓縮資料的位元組數。如果要載入的資料未壓縮，此數值為 0。
lines	integer	此分割要載入的行數。
num_files	integer	此分割要載入的檔案數。
num_files_complete	integer	此分割要載入的檔案數。
current_file	character (256)	此分割要載入的檔案名稱。
pct_complete	integer	此分割已完成載入的資料百分比。

範例查詢

若要檢視 COPY 命令的每個分割的進度，請輸入下列查詢。此範例使用 PG_LAST_COPY_ID() 函式擷取最後一個 COPY 命令的資訊。

```
select slice , bytes_loaded, bytes_to_load , pct_complete from stv_load_state where
 query = pg_last_copy_id();

 slice | bytes_loaded | bytes_to_load | pct_complete
-------+--------------+---------------+--------------
     2 |            0 |             0 |            0
     3 |     12840898 |      39104640 |           32
(2 rows)
```

STV_LOCKS

使用 STV_LOCKS 資料表檢視資料庫中資料表上的任何目前更新。

Amazon Redshift 鎖定資料表以避免兩個使用者同時更新相同的資料表。在 STV_LOCKS 資料表顯示所有目前資料表更新時，查詢 STL_TR_CONFLICT 資料表以查看鎖定衝突的記錄。使用 SVV_TRANSACTIONS 檢視以識別開啟交易與鎖定爭用的問題。

只有超級使用者可以看到 STV_LOCKS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
table_id	bigint	取得鎖定之資料表的資料表 ID。
last_commit	timestamp	資料表中最近一次遞交的時間戳記。
last_update	timestamp	資料表最近一次更新的時間戳記。
lock_owner	bigint	與鎖定關聯的交易 ID。
lock_owner_pid	bigint	與鎖定關聯的處理程序 ID。
lock_owner_start_ts	timestamp	交易開始時的時間戳記。
lock_owner_end_ts	timestamp	交易結束時的時間戳記。
lock_status	character (22)	等待或保持鎖定的處理程序狀態。

範例查詢

若要檢視在目前交易中進行的所有鎖定，請輸入下列命令：

```
select table_id, last_update, lock_owner, lock_owner_pid from stv_locks;
```

此查詢會傳回以下範例輸出，顯示目前生效的三個鎖定：

```
 table_id |        last_update         | lock_owner | lock_owner_pid
```

```
----------+----------------------------+------------+----------------
100004  | 2008-12-23 10:08:48.882319 |       1043 |           5656
100003  | 2008-12-23 10:08:48.779543 |       1043 |           5656
100140  | 2008-12-23 10:08:48.021576 |       1043 |           5656
(3 rows)
```

STV_ML_MODEL_INFO

關於機器學習模型的目前狀態的相關資訊。

所有使用者都可看見 STV_ML_MODEL_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
schema_name	char(128)	模型的命名空間。
user_name	char(128)	模型的擁有者。
model_name	char(128)	模型的名稱。
life_cycle	char(20)	模型的生命週期狀態。
is_refreshable	integer	模型的狀態，指出如果訓練查詢中的原始資料表和欄仍然存在且使用者仍具有這些項目的許可，則模型是否可重新整理。可能的值是：1 (可重新整理) 和 0 (無法重新整理)。
model_state	char(128)	膜性的目前狀態。

範例查詢

下列查詢會顯示機器學習模型的目前狀態。

```
SELECT schema_name, model_name, model_state
FROM stv_ml_model_info;

 schema_name |        model_name            |             model_state
```

```
-------------+------------------------------+--------------------------------------
 public      | customer_churn_auto_model    | Train Model On SageMaker In Progress
 public      | customer_churn_xgboost_model | Model is Ready
(2 row)
```

STV_MV_DEPS

STV_MV_DEPS 資料表顯示了 Amazon Redshift 中具體化視觀表對其他具體化視觀表的相依性。

如需具體化視觀表的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

所有使用者都可看見 STV_MV_DEPS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
db_name	char(128)	包含所指定具體化視觀表的資料庫。
結構描述	char(128)	具體化視觀表的結構描述。
name	char(128)	具體化視觀表的名稱。
ref_schema	char(128)	此具體化視觀表所依賴的具體化視觀表結構描述。
ref_name	char(128)	此具體化視觀表相依之具體化視觀表的名稱。
ref_database_name	char(128)	此具體化視觀表相依的資料庫名稱。

範例查詢

下列查詢會傳回輸出列，指出具體化視觀表 mv_over_foo 使用其定義中的具體化視觀表 mv_foo 做為相依性。

```
CREATE SCHEMA test_ivm_setup;
CREATE TABLE test_ivm_setup.foo(a INT);
CREATE MATERIALIZED VIEW test_ivm_setup.mv_foo AS SELECT * FROM test_ivm_setup.foo;
CREATE MATERIALIZED VIEW test_ivm_setup.mv_over_foo AS SELECT * FROM
 test_ivm_setup.mv_foo;
```

```
SELECT * FROM stv_mv_deps;

 db_name | schema          | name        |   ref_schema   | ref_name |
 ref_database_name
---------+-----------------+-------------+----------------+----------
+------------------
 dev     | test_ivm_setup  | mv_over_foo | test_ivm_setup | mv_foo   | dev
```

STV_MV_INFO

對於每個具體化視觀表，STV_MV_INFO 資料表都會包含一個資料列，顯示資料是否過時，以及狀態資訊。

如需具體化視觀表的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

所有使用者都可看見 STV_MV_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
db_name	char(128)	包含具體化視觀表的資料庫。
結構描述	char(128)	資料庫的結構描述。
name	char(128)	具體化視觀表名稱。
updated_upto_xid	bigint	保留供內部使用。
is_stale	char(1)	t 表示具體化視觀表已過時。「過時」的具體化視觀表是基底資料表已更新，但具體化視觀表尚未重新整理時的具體化視觀表。如果重新整理自上一次重新啟動以來都並未執行，這項資訊可能會不準確。 如果具體化視觀表相依於可變函數，is_stale 欄一律會設定為 t。當給出相同的引數或參數時，可

欄名稱	資料類型	描述
		變函數會傳回不同的結果。例如，傳回日期或時間戳記的大多數函數都是可變函數。
owner_user_name	char(128)	擁有具體化視觀表的使用者。
state	integer	具體化視觀表的狀態如下： • 0 - 重新整理時，會完全重新計算具體化視觀表。 • 1 - 具體化視觀表為累加式。 • 101 - 具體化視觀表因遭到卸除的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 102 - 具體化視觀表因為資料欄類型變更而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 103 - 具體化視觀表因為重新命名的資料表而無法重新整理。 • 104 - 具體化視觀表因為重新命名的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 105 - 具體化視觀表因為重新命名的結構描述而無法重新整理。
autorewrite	char(1)	t 表示具體化視觀表適用於自動重新寫入查詢。
autorefresh	char(1)	t 表示可以自動重新整理具體化視觀表。

範例查詢

若要檢視所有具體化視觀表的狀態，請執行下列查詢。

```
select * from stv_mv_info;
```

此查詢傳回下列範例輸出。

```
 db_name |       schema       |   name   | updated_upto_xid | is_stale | owner_user_name
 | state | autorefresh | autorewrite
---------+--------------------+---------+------------------+----------
+-----------------+-------+-------------+------------
 dev     | test_ivm_setup     | mv       |             1031 | f        | catch-22
 |     1 |           1 |           0
 dev     | test_ivm_setup     | old_mv   |              988 | t        | lotr
 |     1 |           0 |           1
```

STV_NODE_STORAGE_CAPACITY

STV_NODE_STORAGE_CAPACITY 資料表會顯示叢集中每個節點的總儲存容量和使用總容量的詳細資訊。它包含每個節點的一列。

只有超級使用者可以看到 STV_NODE_STORAGE_CAPACITY。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
節點	integer	節點號碼。
used	integer	節點上使用中的 1 MB 磁碟區塊數。對於 RA3 節點類型，使用的區塊包括本機快取的區塊和保存在 Amazon S3 中的區塊。
容量	integer	為節點佈建的總儲存容量 (以 1 MB 為單位)。容量包括由 Amazon Redshift 在 DC2 節點類型上保留的空間，以供內部使用。容量大於名目節點容量，即可用於使用者資料的節點空間量。對於 RA3 節點類型，此容量與叢集的受管理儲存配額總計相同。如需按節點類型分類容量的相關資訊，請參閱《Amazon Redshift 管理指南》中的節點類型詳細資訊。

範例查詢

> **Note**
>
> 下列範例的結果會根據叢集的節點規格而有所不同。將欄 `capacity` 新增至您的 SQL SELECT 以擷取叢集的容量。

下列查詢會傳回 1 MB 磁碟區塊中已使用的空間和總容量。此範例在雙節點 dc2.8xlarge 叢集上執行。

```
select node, used from stv_node_storage_capacity order by node;
```

此查詢傳回下列範例輸出。

```
 node | used
------+-------
    0 | 30597
    1 | 27089
```

下列查詢會傳回 1 MB 磁碟區塊中已使用的空間和總容量。此範例在雙節點 ra3.16xlarge 叢集上執行。

```
select node, used from stv_node_storage_capacity order by node;
```

此查詢傳回下列範例輸出。

```
 node | used
------+-------
    0 | 30591
    1 | 27103
```

STV_PARTITIONS

使用 STV_PARTITIONS 資料表以了解 Amazon Redshift 的磁碟速度效能與磁碟使用率。

STV_PARTITIONS 對於每節點、每邏輯磁碟區皆包含一個資料列。

只有超級使用者可以看到 STV_PARTITIONS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
owner	integer	擁有該分割區的磁碟節點。
託管	integer	實體連接至該分割區的節點。
diskno	integer	包含該分割區的磁碟。
part_begin	bigint	分割區的位移。原始裝置以邏輯分割為鏡像區塊的開放空間。
part_end	bigint	分割區的結尾。
used	integer	分割區上使用中的 1 MB 磁碟區塊數。
tossed	integer	已準備刪除，但因釋放其磁碟地址並不安全而未移除的區塊數。如果立即釋放其地址，等待中的交易可能會寫入磁碟上的相同位置。因此，這些被丟棄的區塊將在下次遞交時釋放。在 INSERT 或磁碟型查詢操作過程中，當資料表欄位被刪除時，磁碟區塊可能會標記為已丟棄。
容量	integer	1 MB 磁碟區塊的分割區總容量。
reads	bigint	從最近一次叢集重新啟動之後發生的讀取次數。
writes	bigint	從最近一次叢集重新啟動之後發生的寫入次數。
seek_forward	integer	非用於後續地址的請求獲得前一個請求地址的次數。
seek_back	integer	非用於前一個地址的請求獲得後續請求地址的次數。
is_san	integer	該分割區是否屬於 SAN。有效值為 **0** (false) 或 **1** (true)。
失敗	integer	此欄位已淘汰。
mbps	integer	磁碟速度，以每秒的 MB 量計算。

欄名稱	資料類型	描述
mount	character (256)	通往裝置的目錄路徑。

範例查詢

以下查詢會傳回已使用的磁碟空間與容量 (以 1 MB 磁碟區塊為單位)，並以原始磁碟空間的百分比計算磁碟使用率。原始磁碟空間包括 Amazon Redshift 保留供內部使用的空間，因此會大於名目磁碟容量，它是可供使用者使用的磁碟空間容量。Amazon Redshift 管理主控台效能索引標籤上的已使用磁碟空間百分比指標會報告叢集使用的標稱磁碟容量百分比。建議您監控 Percentage of Disk Space Used (已使用磁碟空間百分比) 指標，以維護叢集名目磁碟容量的使用情形。

> **Important**
>
> 強烈建議您不要超過叢集的名目磁碟容量。雖然在某些情況下，以技術而言是可能的，但超過名目磁碟容量會降低叢集的容錯能力，同時提高遺失資料的風險。

此範例執行於每節點有六個邏輯磁碟分割區的雙節點叢集。空間的使用非常平均地分散至各磁碟，每個磁碟大約已使用 25%。

```
select owner, host, diskno, used, capacity,
(used-tossed)/capacity::numeric *100 as pctused
from stv_partitions order by owner;

 owner | host | diskno |  used  | capacity | pctused
-------+------+--------+--------+----------+---------
   0   |  0   |    0   | 236480 |  949954  | 24.9
   0   |  0   |    1   | 236420 |  949954  | 24.9
   0   |  0   |    2   | 236440 |  949954  | 24.9
   0   |  1   |    2   | 235150 |  949954  | 24.8
   0   |  1   |    1   | 237100 |  949954  | 25.0
   0   |  1   |    0   | 237090 |  949954  | 25.0
   1   |  1   |    0   | 236310 |  949954  | 24.9
   1   |  1   |    1   | 236300 |  949954  | 24.9
   1   |  1   |    2   | 236320 |  949954  | 24.9
   1   |  0   |    2   | 237910 |  949954  | 25.0
   1   |  0   |    1   | 235640 |  949954  | 24.8
   1   |  0   |    0   | 235380 |  949954  | 24.8
```

```
(12 rows)
```

STV_QUERY_METRICS

包含正在使用者定義的查詢佇列 (服務類別) 中執行之查詢的指標資訊，例如已處理的資料列數目、CPU 用量、輸入/輸出和磁碟使用情形。若要檢視已完成之查詢的指標，請參閱 STL_QUERY_METRICS 系統資料表。

查詢指標每間隔一秒鐘取樣一次。因此，相同查詢的不同執行可能會傳回稍微不同的時間。另外，可能不會記錄執行時間不到一秒的查詢區段。

STV_QUERY_METRICS 會追蹤並彙總查詢、區段和步驟層級的指標。如需查詢區段和步驟的相關資訊，請參閱查詢計劃和執行工作流程。許多指標 (例如 max_rows、cpu_time 等等) 是跨節點配量加總的。如需節點配量的相關資訊，請參閱資料倉儲系統架構。

若要判斷資料列報告指標的層級，請檢查 segment 和 step_type 資料欄：

- 如果 segment 和 step_type 皆為 -1，則資料列報告查詢層級的指標。
- 如果 segment 不是 -1，且 step_type 是 -1，則資料列報告區段層級的指標。
- 如果 segment 和 step_type 皆不是 -1，則資料列報告步驟層級的指標。

所有使用者都可看見 STV_QUERY_METRICS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
service_class	integer	WLM 查詢佇列 ID (服務類別)。查詢佇列定義於 WLM 組態。只會回報使用者定義之佇列的指標。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。

欄名稱	資料類型	描述
starttime	timestamp	查詢開始執行的 UTC 時間，精確度為 6 位數的小數秒。例如：`2009-06-12 11:29:19.131358` 。
slices	integer	叢集的分割數量。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。如果區段值為 -1，指標區段值將累計至查詢層級。
step_type	integer	執行的步驟類型。如需步驟類型的說明，請參閱步驟類型。
rows	bigint	步驟處理的資料列數。
max_rows	bigint	為步驟匯出的最大資料列數 (彙總所有分割)。
cpu_time	bigint	已使用的 CPU 時間，以微秒為單位。在區段層級，所有分割的區段所使用的 CPU 總時間。在查詢層級，所有分割與區段的查詢 CPU 時間總和。
max_cpu_time	bigint	已使用的最大 CPU 時間，以微秒為單位。在區段層級，區段中所有分割已使用的最大 CPU 時間。在查詢層級，任何查詢區段已使用的最大 CPU 時間。
blocks_read	bigint	查詢或區段所讀取的 1 MB 區塊數。
max_blocks_read	bigint	區段讀取的 1 MB 區塊最大數量 (彙總所有分割)。在區段層級，所有分割的區段所讀取的 1 MB 區塊最大數量。在查詢層級，任何查詢區段所讀取的 1 MB 區塊最大數量。
run_time	bigint	所有分割合計的總執行時間。執行時間不包含等待時間。 在區段層級，所有分割合計的區段執行時間。在查詢層級，所有分割與區段的查詢執行時間總和。因為此值為總和，因此執行時間與查詢執行時間無關。
max_run_time	bigint	區段的最大經過時間 (以微秒為單位)。在區段層級，所有分割之區段的最大執行時間。在查詢層級，任何查詢區段的最大執行時間。

欄名稱	資料類型	描述
max_blocks_to_disk	bigint	用於寫入中繼結果的磁碟空間最大數量，以 1 MB 區塊為單位。在區段層級，所有分割之區段已使用的最大磁碟空間量。在查詢層級，任何查詢區段已使用的最大磁碟空間量。
blocks_to_disk	bigint	查詢或區段用於寫入中繼結果的最大磁碟空間量，以 1 MB 區塊為單位。
step	integer	執行的查詢步驟。
max_query_scan_size	bigint	查詢所掃描的最大資料大小，以 MB 為單位。在區段層級，所有分割之區段所掃描的最大資料大小。在查詢層級，任何查詢區段所掃描的最大資料大小。
query_scan_size	bigint	查詢所掃描的資料大小，以 MB 為單位。
query_priority	integer	查詢的優先順序。可能的值為 -1、0、1、2、3 和 4，其中 -1 表示不支援查詢優先順序。
query_queue_time	bigint	查詢排入佇列的時間 (毫秒)。

步驟類型

下表列出與資料庫使用者相關的步驟類型。此表未列出僅供內部使用的步驟類型。如果步驟類型為 -1，則該步驟層級不會回報該指標。

Step type (步驟類型)	描述
1	掃描資料表
2	插入資料列
3	彙總資料列
6	排序步驟
7	合併步驟

Step type (步驟類型)	描述
8	分佈步驟
9	廣播分佈步驟
10	雜湊聯結
11	合併聯結
12	儲存步驟
14	Hash
15	巢狀迴路聯結
16	投射欄位與表達式
17	限制傳回的資料列數
18	唯一
20	刪除資料列
26	限制傳回的排序資料列數
29	運算視窗函式
32	UDF
33	唯一
37	從領導者節點將資料列傳回至用戶端
38	從運算節點將資料列傳回至領導節點
40	Spectrum 掃描。

範例查詢

若要找出具有高 CPU 時間 (超過 1,000 秒) 的使用中查詢，請執行下列查詢。

```
select query, cpu_time / 1000000 as cpu_seconds
from stv_query_metrics where segment = -1 and cpu_time > 1000000000
order by cpu_time;

query | cpu_seconds
------+------------
25775 |        9540
```

若要找出具有傳回超過一百萬個資料列之巢狀迴圈聯結的使用中查詢，請執行下列查詢。

```
select query, rows
from stv_query_metrics
where step_type = 15 and rows > 1000000
order by rows;

query | rows
------+-----------
25775 | 1580225854
```

若要找出已執行超過 60 秒，但已使用 CPU 時間少於 10 秒的使用中查詢，請執行下列查詢。

```
select query, run_time/1000000 as run_time_seconds
from stv_query_metrics
where segment = -1 and run_time > 60000000 and cpu_time < 10000000;

query | run_time_seconds
------+-----------------
25775 |              114
```

STV_RECENTS

使用 STV_RECENTS 資料表找出目前使用中並在最近針對資料庫執行之查詢的相關資訊。

所有使用者都可看見 STV_RECENTS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

Troubleshooting with STV_RECENTS

STV_RECENTS 對於確定查詢或查詢集合目前是否正在執行或已完成特別有用。它也會顯示查詢執行的持續時間。這對於了解哪些查詢長時間執行很有幫助。

您可以將 STV_RECENTS 聯結至其他系統檢視，例如 STV_INFLIGHT，以收集有關執行查詢的其他中繼資料。(範例查詢部分中有一個範例展示如何執行此操作。) 您也可以使用此檢視中的傳回記錄以及 Amazon Redshift 主控台中的監控功能，進行即時疑難排解。

補充 STV_RECENTS 的系統檢視包括 STL_QUERYTEXT (會擷取 SQL 命令的查詢文字)，以及 SVV_QUERY_INFLIGHT (會將 STV_INFLIGHT 連接到 STL_QUERYTEXT)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
status	character(20)	查詢狀態。有效值為 **Running**、**Done**。
starttime	timestamp	查詢開始的時間。
持續時間	integer	工作階段開始之後的時間 (微秒)。
user_name	character(50)	執行處理程序者的使用者名稱。
db_name	character(50)	資料庫的名稱。
query	character(600)	查詢文字，最多 600 個字元。任何額外的字元都會被截斷。
pid	integer	與查詢關聯之工作階段的處理程序 ID，已完成之查詢的此 ID 一律為 -1。

範例查詢

若要判斷目前正在資料庫上執行的有哪些查詢，請執行下列查詢：

```
select user_name, db_name, pid, query
from stv_recents
where status = 'Running';
```

以下範例輸出顯示執行於 TICKIT 資料庫的單一查詢：

```
user_name | db_name |   pid   | query
----------+---------+---------+-------------
dwuser    | tickit  |  19996  |select venuename, venueseats from
venue where venueseats > 50000 order by venueseats desc;
```

下列範例傳回正在執行或在佇列中等待執行之查詢 (如果有) 的清單：

```
select * from stv_recents where status<>'Done';

status |    starttime        | duration |user_name|db_name| query     | pid
-------+---------------------+----------+---------+-------+-----------+------
Running| 2010-04-21 16:11... | 281566454| dwuser  |tickit | select ...| 23347
```

除非您正在執行多個並行查詢，而且其中部分查詢位於佇列中，否則此查詢不會傳回結果。

以下範例延伸之前的範例。在此情況中，真正「進行中」(執行中而非等待中) 的查詢會排除在結果之外：

```
select * from stv_recents where status<>'Done'
and pid not in (select pid from stv_inflight);
...
```

如需疑難排解查詢效能的詳細提示，請參閱對查詢進行故障診斷。

STV_SESSIONS

使用 STV_SESSIONS 資料表以檢視 Amazon Redshift 的作用中使用者工作階段的相關資訊。

若要檢視工作階段歷程記錄，請使用 STL_SESSIONS 資料表而非 STV_SESSIONS。

所有使用者都可看見 STV_SESSIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_SESSION_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
starttime	timestamp	工作階段開始的時間。
process	integer	工作階段的處理程序 ID。
user_name	character(50)	與工作階段相關聯的使用者。
db_name	character(50)	與工作階段相關聯之資料庫的名稱。
timeout_sec	int	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。

範例查詢

若要執行快速檢查以查看是否有任何其他使用者目前已登入 Amazon Redshift，請輸入下列查詢：

```
select count(*)
from stv_sessions;
```

如果結果大於 1，則至少有一位其他使用者目前已登入資料庫。

若要檢視 Amazon Redshift 的所有使用中工作階段，請輸入下列查詢：

```
select *
from stv_sessions;
```

以下結果顯示 Amazon Redshift 上目前執行的四個作用中工作階段：

```
       starttime          | process |user_name                    | db_name
      | timeout_sec
--------------------------+---------+-----------------------------
+---------------------------+-------------
  2018-08-06 08:44:07.50 |   13779 | IAMA:aws_admin:admin_grp    | dev
      | 0
  2008-08-06 08:54:20.50 |   19829 | dwuser                      | dev
      | 120
```

```
  2008-08-06 08:56:34.50 |   20279 | dwuser                    | dev
      | 120
  2008-08-06 08:55:00.50 |   19996 | dwuser                    | tickit
      | 0
(3 rows)
```

字首為 IAMA 的使用者名稱指出使用者是使用聯合單一登入方式登入。如需詳細資訊，請參閱使用 IAM 身分驗證產生資料庫使用者登入資料。

STV_SLICES

使用 STV_SLICES 資料表以檢視目前分割至節點的映射。

STV_SLICES 中的資訊主要用於調查目的。

所有使用者都可看見 STV_SLICES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
節點	integer	分割所在的叢集節點。
分割	integer	節點分割。
localslice	integer	此資訊僅供內部使用。
type	character (1)	此資訊僅供內部使用。

範例查詢

若要檢視哪些叢集節點映射哪些分割，請輸入下列查詢：

```
select node, slice from stv_slices;
```

此查詢傳回下列範例輸出：

```
 node | slice
------+-------
    0 |     2
    0 |     3
    0 |     1
    0 |     0
(4 rows)
```

STV_STARTUP_RECOVERY_STATE

記錄在叢集重新啟動操作期間暫時鎖定的資料表狀態。Amazon Redshift 會在處理資料表時暫時鎖定資料表，以解決叢集重新啟動後的過時交易。

所有使用者都可看見 STV_STARTUP_RECOVERY_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
db_id	integer	資料庫 ID。
table_id	integer	表格 ID。
table_name	character(137)	資料表名稱

範例查詢

若要監控有哪些資料表被暫時鎖定，請在叢集重新啟動之後，執行以下查詢。

```
select * from STV_STARTUP_RECOVERY_STATE;

  db_id | tbl_id | table_name
--------+--------+------------
 100044 | 100058 | lineorder
 100044 | 100068 | part
 100044 | 100072 | customer
 100044 | 100192 | supplier
(4 rows)
```

STV_TBL_PERM

STV_TBL_PERM 資料表包含 Amazon Redshift 中永久資料表的相關資訊，其中包括使用者為目前工作階段建立的暫時資料表。STV_TBL_PERM 包含所有資料庫中所有資料表的資訊。

此資料表不同於 STV_TBL_TRANS，它包含系統在查詢處理過程中建立之暫時性資料庫的相關資訊。

只有超級使用者可以看到 STV_TBL_PERM。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
分割	integer	配置至資料表的節點分割。
id	integer	表格 ID。
name	character (72)	資料表名稱.
rows	bigint	分割中的資料列數。
sorted_rows	bigint	已在磁碟上排序之分割中的資料列數。如果此數字不符合 ROWS 數字，請清空資料表以重新排序資料列。
temp	integer	該資料表是否為暫時資料表。0 = false；1 = true。
db_id	integer	用於建立資料表之資料庫的 ID。
insert_pristine	integer	供內部使用。
delete_pristine	integer	供內部使用。
backup	integer	用於指出該資料表是否包含在叢集快照中的值。0 = no；1 = yes。如需詳細資訊，請參閱 CREATE TABLE 命令的 BACKUP 參數。
dist_style	integer	切片所屬資料表的分散樣式。如需這些值的詳細資訊，請參閱檢視分佈樣式。如需分散樣式的資訊，請參閱分佈樣式。

欄名稱	資料類型	描述
block_count	integer	分割使用的區塊數目。無法計算區塊計數時，值為 -1。

範例查詢

下列查詢會傳回不同的資料表 ID 與名稱的清單：

```
select distinct id, name
from stv_tbl_perm order by name;

   id   |          name
--------+-------------------------
 100571 | category
 100575 | date
 100580 | event
 100596 | listing
 100003 | padb_config_harvest
 100612 | sales
...
```

其他系統資料表使用資料表 ID，因此知道哪個資料表 ID 對應至某特定資料表是非常有用的。在此範例中，SELECT DISTINCT 用於移除重複 (資料表被發佈至多個分割)。

若要判斷 VENUE 資料表中各個欄位使用的區塊數，請輸入以下查詢：

```
select col, count(*)
from stv_blocklist, stv_tbl_perm
where stv_blocklist.tbl = stv_tbl_perm.id
and stv_blocklist.slice = stv_tbl_perm.slice
and stv_tbl_perm.name = 'venue'
group by col
order by col;

 col | count
-----+-------
   0 |     8
   1 |     8
   2 |     8
   3 |     8
```

```
   4 |     8
   5 |     8
   6 |     8
   7 |     8
(8 rows)
```

使用須知

ROWS 欄位包含未清空但已刪除 (或已清空但有 SORT ONLY 選項) 的資料列數。因此，當您直接查詢特定資料表時，STV_TBL_PERM 資料表中 ROWS 欄位的總和 (SUM) 可能會與 COUNT(*) 結果不相符。例如，如果從 VENUE 刪除 2 個資料列，COUNT(*) 結果為 200，但 SUM(ROWS) 結果仍是 202：

```
delete from venue
where venueid in (1,2);

select count(*) from venue;
count
-------
200
(1 row)

select trim(name) tablename, sum(rows)
from stv_tbl_perm where name='venue' group by name;

tablename | sum
-----------+-----
venue     | 202
(1 row)
```

為了同步 STV_TBL_PERM 中的資料，請執行完整清空 VENUE 資料表。

```
vacuum venue;

select trim(name) tablename, sum(rows)
from stv_tbl_perm
where name='venue'
group by name;

tablename | sum
-----------+-----
venue     | 200
```

```
(1 row)
```

STV_TBL_TRANS

使用 STV_TBL_TRANS 資料表以尋找目前在記憶體中的暫時性資料庫資料表的相關資訊。

暫時性資料表通常是暫存的資料列集，在查詢執行時做為中繼結果使用。STV_TBL_TRANS 不同於 STV_TBL_PERM，STV_TBL_PERM 包含永久資料庫資料表的相關資訊。

只有超級使用者可以看到 STV_TBL_TRANS。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
分割	integer	配置至資料表的節點分割。
id	integer	表格 ID。
rows	bigint	資料表中的資料列數。
size	bigint	配置至資料表的位元組數。
query_id	bigint	查詢 ID。
ref_cnt	integer	參考的數量。
from_suspended	integer	此資料表是否是在查詢 (目前已停用) 期間內所建立的。
prep_swap	integer	此暫時性資料表是否已準備好在需要時交換到磁碟。(此交換只會發生在記憶體量較低的情況。)

範例查詢

若要檢視查詢 ID 為 90 之查詢的暫時性資料表資訊，請輸入以下命令：

```
select slice, id, rows, size, query_id, ref_cnt
from stv_tbl_trans
where query_id = 90;
```

此查詢會傳回查詢 90 的暫時性資料表資訊，如下範例輸出所示：

```
slice | id | rows | size | query_ | ref_ | from_     | prep_
      |    |      |      | id     | cnt  | suspended | swap
------+----+------+------+--------+------+-----------+-------
 1013 | 95 |    0 |    0 |     90 |    4 |         0 |   0
    7 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   10 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   17 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   14 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    3 | 96 |    0 |    0 |     90 |    4 |         0 |   0
 1013 | 99 |    0 |    0 |     90 |    4 |         0 |   0
    9 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    5 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   19 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    2 | 96 |    0 |    0 |     90 |    4 |         0 |   0
 1013 | 98 |    0 |    0 |     90 |    4 |         0 |   0
   13 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    1 | 96 |    0 |    0 |     90 |    4 |         0 |   0
 1013 | 96 |    0 |    0 |     90 |    4 |         0 |   0
    6 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   11 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   15 | 96 |    0 |    0 |     90 |    4 |         0 |   0
   18 | 96 |    0 |    0 |     90 |    4 |         0 |   0
```

在此範例中，您可以看到涉及資料表 95、96 及 98 的查詢資料。因為零字元組分配給此資料表，所以此查詢可在記憶體中執行。

STV_WLM_CLASSIFICATION_CONFIG

包含 WLM 目前的分類規則。

只有超級使用者可以看到 STV_WLM_CLASSIFICATION_CONFIG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
id	integer	服務類別 ID。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。

欄名稱	資料類型	描述
condition	character(128)	查詢條件。
action_seq	integer	保留以供系統使用。
動作	character(64)	保留以供系統使用。
action_service_class	integer	動作發生時所在的服務類別。

範例查詢

```
select * from STV_WLM_CLASSIFICATION_CONFIG;

id | condition                                    | action_seq | action |
 action_service_class
---+----------------------------------------------+------------+--------
+---------------------
 1 | (system user) and (query group: health)     |          0 | assign |
     1
 2 | (system user) and (query group: metrics)    |          0 | assign |
     2
 3 | (system user) and (query group: cmstats)    |          0 | assign |
     3
 4 | (system user)                               |          0 | assign |
     4
 5 | (super user) and (query group: superuser)   |          0 | assign |
     5
 6 | (query group: querygroup1)                  |          0 | assign |
     6
 7 | (user group: usergroup1)                    |          0 | assign |
     6
 8 | (user group: usergroup2)                    |          0 | assign |
     7
 9 | (query group: querygroup3)                  |          0 | assign |
     8
10 | (query group: querygroup4)                  |          0 | assign |
     9
11 | (user group: usergroup4)                    |          0 | assign |
     9
```

```
12 | (query group: querygroup*)                    |          0 | assign |
    10
13 | (user group: usergroup*)                      |          0 | assign |
    10
14 | (querytype: any)                              |          0 | assign |
    11
(4 rows)
```

STV_WLM_QMR_CONFIG

記錄 WLM 查詢監控規則 (QMR) 的組態設定。如需詳細資訊，請參閱 WLM 查詢監控規則。

只有超級使用者可以看到 STV_WLM_QMR_CONFIG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
service_class	integer	WLM 查詢佇列 ID (服務類別)。查詢佇列定義於 WLM 組態。只有使用者定義的佇列可以定義規則。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
rule_name	character(256)	查詢監控規則的名稱。
動作	character(256)	規則動作。可能值為 log、hop、abort 和 change_query_priority 。
metric_name	character(256)	指標的名稱。
metric_operator	character(256)	指標運算子。可能值為 >、=、<。
metric_value	double	觸發動作之指定指標的閾值。
action_value	character(256)	若 action 為 change_query_priority ，則可能的值會是 highest、high、normal、low 和 lowest。 若 action 為 log、hop 或 abort，則此值為是空白。

範例查詢

若要檢視所有大於 5 之服務類別 (包括使用者定義的佇列) 的 QMR 規則定義，請執行以下查詢。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。

```
Select *
from stv_wlm_qmr_config
where service_class > 5
order by service_class;
```

STV_WLM_QUERY_QUEUE_STATE

記錄服務類別之查詢佇列的目前狀態。

所有使用者都可看見 STV_WLM_QUERY_QUEUE_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
position	integer	佇列中查詢的位置。下一個執行的是擁有最小 **position** 值的查詢。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可與多個查詢 ID 關聯。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
slot_count	integer	WLM 查詢插槽的數量。
start_time	timestamp	查詢進入佇列的時間。

欄名稱	資料類型	描述
queue_time	bigint	查詢在佇列中的時間 (微秒)。

範例查詢

下列查詢會顯示服務類別大於 4 的佇列中的查詢。

```
select * from stv_wlm_query_queue_state
where service_class > 4
order by service_class;
```

此查詢傳回下列範例輸出。

```
 service_class | position | task | query | slot_count |        start_time          |
 queue_time
---------------+----------+------+-------+------------+----------------------------
+------------
             5 |        0 |  455 |   476 |          5 | 2010-10-06 13:18:24.065838 |
 20937257
             6 |        1 |  456 |   478 |          5 | 2010-10-06 13:18:26.652906 |
 18350191
(2 rows)
```

STV_WLM_QUERY_STATE

記錄 WLM 追蹤之查詢的目前狀態。

所有使用者都可看見 STV_WLM_QUERY_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
xid	integer	查詢或子查詢的交易 ID。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可與多個查詢 ID 關聯。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
service_class	integer	服務類別的 ID。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
slot_count	integer	WLM 查詢插槽的數量。
wlm_start_time	timestamp	查詢進入系統資料表佇列或短期查詢佇列的時間。
state	character(16)	查詢或子查詢的目前狀態。 可能的值如下： • Classified - 查詢已指派給服務類別。 • Completed - 查詢已完成執行。查詢執行成功或已取消。對於最終狀態，請檢查 STL_QUERY 的結果。 • Dequeued – 僅供內部使用。 • Evicted - 已從服務類別撤銷查詢以重新啟動。 • Evicting - 正從服務類別撤銷查詢以重新啟動。 • Initialized – 僅供內部使用。 • Invalid – 僅供內部使用。 • Queued - 查詢已傳送至查詢佇列，因為沒有可用的插槽可用來執行查詢佇列。 • QueuedWaiting - 查詢正在查詢佇列中等待。 • Rejected – 僅供內部使用。

欄名稱	資料類型	描述
		• Returning - 查詢正在將結果傳回給用戶端。 • Running - 正在執行查詢。 • TaskAssigned – 僅供內部使用。
queue_time	bigint	查詢在佇列中花費的時間 (微秒)。
exec_time	bigint	查詢已執行的微秒數。
query_priority	char(20)	查詢的優先順序。可能的值為 n/a、lowest、low、normal、high 和 highest，其中 n/a 表示不支援查詢優先順序。

範例查詢

下列查詢顯示大於 4 之服務類別中所有目前正在執行的查詢。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。

```
select xid, query, trim(state) as state, queue_time, exec_time
from stv_wlm_query_state
where service_class > 4;
```

此查詢傳回下列範例輸出：

```
xid    | query | state   | queue_time | exec_time
-------+-------+---------+------------+-----------
100813 | 25942 | Running |          0 |    1369029
100074 | 25775 | Running |          0 | 2221589242
```

STV_WLM_QUERY_TASK_STATE

包含服務類別查詢任務的目前狀態。

所有使用者都可看見 STV_WLM_QUERY_TASK_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可與多個查詢 ID 關聯。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
slot_count	integer	WLM 查詢插槽的數量。
start_time	timestamp	查詢開始執行的時間。
exec_time	bigint	查詢到目前為止已執行的時間 (微秒)。

範例查詢

下列查詢顯示大於 4 之服務類別中的查詢的目前狀態。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。

```
select * from stv_wlm_query_task_state
where service_class > 4;
```

此查詢傳回下列範例輸出：

```
service_class | task | query |         start_time         | exec_time
--------------+------+-------+----------------------------+-----------
    5         |  466 |   491 | 2010-10-06 13:29:23.063787 | 357618748
(1 row)
```

STV_WLM_SERVICE_CLASS_CONFIG

記錄 WLM 的服務類別組態。

只有超級使用者可以看到 STV_WLM_SERVICE_CLASS_CONFIG。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
queueing_strategy	character(32)	保留以供系統使用。
num_query_tasks	integer	服務類別的目前實際並行層級。如果 num_query_tasks 與 target_num_query_tasks 不同，表示動態 WLM 轉換正在進行中。值 -1 表示已設定 Auto WLM (自動 WLM)。
target_num_query_tasks	integer	由最新的 WLM 組態變更所設定的並行層級。
evictable	character(8)	保留以供系統使用。
eviction_threshold	bigint	保留以供系統使用。
query_working_mem	integer	指派至服務類別的目前實際運作中記憶體數量，以每節點、每插槽 MB 為單位。如果 query_working_mem 與 target_query_working_mem 不同，表示動態 WLM 轉換正在進行中。值 -1 表示已設定 Auto WLM (自動 WLM)。
target_query_working_mem	integer	由最新 WLM 組態變更設定的運作中記憶體數量，以每節點、每插槽 MB 為單位。
min_step_mem	integer	保留以供系統使用。
name	character(64)	服務類別的名稱。
max_execution_time	bigint	查詢在被終止之前可執行的時間 (微秒)。

欄名稱	資料類型	描述
user_group_wild_card	Boolean	如果為 TRUE，WLM 佇列會將星號 (*) 視為 WLM 組態中使用者群組字串中的萬用字元。
query_group_wild_card	Boolean	如果為 TRUE，WLM 佇列會將星號 (*) 視為 WLM 組態中查詢群組字串中的萬用字元。
concurrency_scaling	character(20)	描述並行擴展是 on 或 off。
query_priority	character(20)	查詢優先順序的值。
user_role_wild_card	Boolean	如果為 TRUE，WLM 佇列會將星號 (*) 視為 WLM 組態中使用者字串中的萬用字元。

範例查詢

第一個使用者定義的服務類別為服務類別 6，其名為 Service class #1。下列查詢顯示大於 4 之服務類別的目前組態。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。

```
select rtrim(name) as name,
num_query_tasks as slots,
query_working_mem as mem,
max_execution_time as max_time,
user_group_wild_card as user_wildcard,
query_group_wild_card as query_wildcard
from stv_wlm_service_class_config
where service_class > 4;

name                         | slots | mem | max_time | user_wildcard | query_wildcard
-----------------------------+-------+-----+----------+---------------+---------------
Service class for super user |     1 | 535 |        0 | false         | false
Queue 1                      |     5 | 125 |        0 | false         | false
Queue 2                      |     5 | 125 |        0 | false         | false
Queue 3                      |     5 | 125 |        0 | false         | false
Queue 4                      |     5 | 627 |        0 | false         | false
Queue 5                      |     5 | 125 |        0 | true          | true
Default queue                |     5 | 125 |        0 | false         | false
```

以下查詢顯示動態 WLM 轉換的狀態。在轉換進行時，num_query_tasks 與 target_query_working_mem 會進行更新，直到它們等於目標值。如需詳細資訊，請參閱 WLM 動態和靜態組態屬性。

```
select rtrim(name) as name,
num_query_tasks as slots,
target_num_query_tasks as target_slots,
query_working_mem as memory,
target_query_working_mem as target_memory
from stv_wlm_service_class_config
where num_query_tasks > target_num_query_tasks
or query_working_mem > target_query_working_mem
and service_class > 5;

 name              | slots | target_slots | memory | target_mem
-------------------+-------+--------------+--------+------------
 Queue 3           |     5 |           15 |    125 |        375
 Queue 5           |    10 |            5 |    250 |        125
 (2 rows)
```

STV_WLM_SERVICE_CLASS_STATE

包含服務類別的目前狀態。

只有超級使用者可以看到 STV_WLM_SERVICE_CLASS_STATE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
num_queued_queries	integer	目前佇列中的查詢數。
num_executing_queries	integer	目前執行中的查詢數。
num_serviced_queries	integer	曾經在服務類別中的查詢數。

欄名稱	資料類型	描述
num_executed_queries	integer	自 Amazon Redshift 重新啟動後執行的查詢數目。
num_evicted_queries	integer	在 Amazon Redshift 重新啟動之後已移出的查詢數量。查詢被移出的部分原因包括 WLM 逾時、QMR 跳轉動作，以及在並行擴展叢集上查詢失敗。
num_concurrency_scaling_queries	integer	在 Amazon Redshift 重新啟動之後執行於並行擴展叢集的查詢數量。

範例查詢

下列查詢顯示大於 5 之服務類別的狀態。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。

```
select service_class, num_executing_queries,
num_executed_queries
from stv_wlm_service_class_state
where service_class > 5
order by service_class;
```

```
 service_class | num_executing_queries | num_executed_queries
---------------+-----------------------+----------------------
             6 |                     1 |                  222
             7 |                     0 |                  135
             8 |                     1 |                   39
(3 rows)
```

STV_X 還原 _ 交換佇列狀態

在傳統調整大小期間，使用 STV_XRESTORE 來監視每個資料表的移轉進度。當目標節點類型為 RA3 時，這特別適用。有關經典調整為 RA3 節點的更多信息，請轉到經典調整大小。

只有超級使用者才能看到 STV_XRESTORE _ 交換隊列 _ 狀態。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_RESTORE_STATE 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	起始調整大小的使用者 ID。
db_id	integer	資料庫的識別碼。
結構描述	char(128)	結構描述的名稱。
table_name	char(128)	資料表的名稱。
tbl	integer	資料表的 ID。
status	char(64)	資料表的移轉進度狀態。可能的值如下。 • Waiting：等待重新發佈開始 • Applying：目前重新分配 • Finished: 已完成的重新分配
任務類型	integer	資料表的重新發佈類型。可能的值如下。 • 1：鍵 • 2：甚至 如需分配型式的更多資訊，請參閱〈〉分佈樣式。

範例查詢

下列查詢顯示資料庫中正在等待調整大小、目前正在調整大小以及完成重新調整大小的資料表數目。

```
select db_id, status, count(*)
from stv_xrestore_alter_queue_state
group by 1,2 order by 3 desc

db_id  |   status    | count
-------+-------------+------
694325 | Waiting     |   323
694325 | Finished    |    60
```

```
694325 | Applying   |     1
```

主要和並行擴展叢集的 SVCS 檢視

字首為 SVCS 的 SVCS 系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 STL 的資料表，差別在於 STL 資料表僅提供執行於主要叢集之查詢的資訊。

主題

- SVCS_ALERT_EVENT_LOG
- SVCS_COMPILE
- SVCS_CONCURRENCY_SCALING_USAGE
- SVCS_EXPLAIN
- SVCS_PLAN_INFO
- SVCS_QUERY_SUMMARY
- SVCS_S3LIST
- SVCS_S3LOG
- SVCS_S3PARTITION_SUMMARY
- SVCS_S3QUERY_SUMMARY
- SVCS_STREAM_SEGS
- SVCS_UNLOAD_LOG

SVCS_ALERT_EVENT_LOG

當查詢最佳化器識別可能表示效能問題的狀況時，請記錄一個提醒。此檢視衍生自 STL_ALERT_EVENT_LOG 系統資料表，但不會顯示執行於並行擴展叢集之查詢的分割層級。使用 SVCS_ALERT_EVENT_LOG 資料表來識別提升查詢效能的機會。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱查詢處理。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 STL 的資料表，差別在於 STL 資料表僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_ALERT_EVENT_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
pid	integer	與陳述式和配量相關聯的處理程序 ID。如果相同的查詢在多個配量上執行，則其可能具有多個 PID。
xid	bigint	與陳述式關聯的交易 ID。
事件	character (1024)	提醒事件的說明。
solution	character (1024)	建議的解決方案。
event_time	timestamp	查詢開始的時間，以 UTC 表示。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如：**2009-06-12 11:29:19.131358** 。

使用須知

您可以使用 SVCS_ALERT_EVENT_LOG，識別查詢中的潛在問題，然後遵循調校查詢效能中的實務，來最佳化資料庫設計並重新撰寫查詢。SVCS_ALERT_EVENT_LOG 會記錄下列提醒：

- 找不到統計資訊

  找不到統計資訊。在進行資料載入或重要更新之後執行 ANALYZE，並使用 STATUPDATE 與 COPY 操作搭配。如需詳細資訊，請參閱設計查詢的 Amazon Redshift 最佳實務。
- 巢狀迴圈

巢狀迴路通常是 Cartesian 產品。評估您的查詢，以確保所有參與資料表均已有效聯結。

- 選擇性相當高的篩選條件

  傳回的資料列與已掃描資料列的比率低於 0.05。已掃描資料列是 rows_pre_user_filter 的值，而傳回的資料列則是 STL_SCAN 系統資料表中的資料列值。表示查詢正在掃描異常大量的資料列來決定結果集。這可能是由於找不到排序索引鍵或其不正確所致。如需詳細資訊，請參閱使用排序索引鍵。

- 過多的幽靈資料列

  掃描已略過相當多標示為已刪除但未清空的資料列，或已插入但未遞交的資料列。如需詳細資訊，請參閱清空資料表。

- 大型分佈

  已重新配送超過 1,000,000 個資料列，進行雜湊聯結或彙整。如需詳細資訊，請參閱使用資料分佈樣式。

- 大型廣播

  已播送超過 1,000,000 個資料列，進行雜湊聯結。如需詳細資訊，請參閱使用資料分佈樣式。

- 序列執行

  已在查詢計劃中指出 DS_DIST_ALL_INNER 重新配送樣式，其會強制序列執行，因為整個內部資料表已重新配送至單一節點。如需詳細資訊，請參閱使用資料分佈樣式。

範例查詢

下列查詢顯示四個查詢的提醒事件。

```
SELECT query, substring(event,0,25) as event,
substring(solution,0,25) as solution,
trim(event_time) as event_time from svcs_alert_event_log order by query;

 query |             event             |          solution            |     event_time

-------+-------------------------------+-------------------------------
+---------------------
  6567 | Missing query planner statist | Run the ANALYZE command       | 2014-01-03
 18:20:58
  7450 | Scanned a large number of del | Run the VACUUM command to rec| 2014-01-03
 21:19:31
```

```
  8406 | Nested Loop Join in the query | Review the join predicates to| 2014-01-04
 00:34:22
 29512 | Very selective query filter:r | Review the choice of sort key| 2014-01-06
 22:00:00

(4 rows)
```

SVCS_COMPILE

記錄查詢之每個查詢區段的編譯時間和位置，包括執行於擴展叢集的查詢以及執行於主要叢集的查詢。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 SVL 的檢視，差別在於 SVL 檢視僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_COMPILE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

如需 SCL_COMPILE 的相關資訊，請參閱 SVL_COMPILE。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。
query	integer	查詢 ID。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
segment	integer	要編譯的查詢區段。
locus	integer	執行區段的位置，**1** (若在運算節點上) 和 **2** (若在領導節點上)。
starttime	timestamp	編譯開始的時間，以國際標準時間 (UTC) 表示。

欄名稱	資料類型	描述
endtime	timestamp	編譯結束的時間，以 UTC 表示。
compile	integer	值為 **0** 表示已重複使用編譯，**1** 表示已編譯區段。

範例查詢

在此範例中，查詢 35878 和 35879 執行相同的 SQL 陳述式。查詢 35878 的編譯欄位針對四個查詢區段顯示 1，此值表示該區段已編譯。查詢 35879 在每個區段的編譯欄位中顯示 0，表示該區段不需要重新編譯。

```
select userid, xid,  pid, query, segment, locus,
datediff(ms, starttime, endtime) as duration, compile
from svcs_compile
where query = 35878 or query = 35879
order by query, segment;

 userid |  xid   |  pid  | query | segment | locus | duration | compile
--------+--------+-------+-------+---------+-------+----------+---------
    100 | 112780 | 23028 | 35878 |       0 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       1 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       2 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       3 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       4 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       5 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       6 |     1 |     1380 |       1
    100 | 112780 | 23028 | 35878 |       7 |     1 |     1085 |       1
    100 | 112780 | 23028 | 35878 |       8 |     1 |     1197 |       1
    100 | 112780 | 23028 | 35878 |       9 |     2 |      905 |       1
    100 | 112782 | 23028 | 35879 |       0 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       1 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       2 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       3 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       4 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       5 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       6 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       7 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       8 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       9 |     2 |        0 |       0
```

```
(20 rows)
```

SVCS_CONCURRENCY_SCALING_USAGE

記錄並行擴展的用量期間。每個用量期間都是連續的持續時間，個別並行擴展叢集在此期間主動處理查詢。

SVCS_CONCURRENCY_SCALING_USAGE 超級使用者可以看到此表格。資料庫超級使用者可選擇將它開啟供所有使用者檢視。

資料表欄

欄名稱	資料類型	描述
start_time	沒有時區的時間戳記	用量期間開始的時間。
end_time	沒有時區的時間戳記	用量期間結束的時間。
queries	bigint	在此用量期間執行的查詢數量。
usage_in_seconds	numeric(27,0)	此用量期間的總秒數。

範例查詢

若要以秒為單位檢視特定期間的用量持續時間，請輸入以下查詢：

```
select * from svcs_concurrency_scaling_usage order by start_time;

start_time | end_time | queries | usage_in_seconds
---------------------------+---------------------------+---------+------------------
2019-02-14 18:43:53.01063 | 2019-02-14 19:16:49.781649 | 48 | 1977
```

SVCS_EXPLAIN

顯示已提交供執行之用的查詢的 EXPLAIN 計劃。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 STL 的資料表，差別在於 STL 資料表僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_EXPLAIN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別符，其中節點會在查詢執行時映射至一個或多個步驟。
parentid	integer	父節點的計劃節點識別符。父節點具有一些子節點。例如，合併聯結是已聯結資料表上掃描的父項。
plannode	character(400)	來自 EXPLAIN 輸出的節點文字。在 EXPLAIN 輸出中，提及在運算節點上執行的計劃節點是以 **XN** 為字首。
info	character(400)	計劃節點的限定詞和篩選條件資訊。例如，此資料欄中包括聯結條件和 WHERE 子句限制。

範例查詢

考慮彙總聯結查詢的下列 EXPLAIN 輸出：

```
explain select avg(datediff(day, listtime, saletime)) as avgwait
from sales, listing where sales.listid = listing.listid;
                                QUERY PLAN

------------------------------------------------------------------------------
 XN Aggregate  (cost=6350.30..6350.31 rows=1 width=16)
```

```
  ->  XN Hash Join DS_DIST_NONE  (cost=47.08..6340.89 rows=3766 width=16)
        Hash Cond: ("outer".listid = "inner".listid)
        -> XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=12)
        -> XN Hash  (cost=37.66..37.66 rows=3766 width=12)
              -> XN Seq Scan on sales  (cost=0.00..37.66 rows=3766 width=12)
(6 rows)
```

如果執行此查詢且其查詢 ID 為 10，則您可以使用 SVCS_EXPLAIN 資料表，來查看 EXPLAIN 命令所傳回相同類型的資訊：

```
select query,nodeid,parentid,substring(plannode from 1 for 30),
substring(info from 1 for 20) from svcs_explain
where query=10 order by 1,2;

query| nodeid |parentid|           substring            |    substring
-----+--------+--------+--------------------------------+-------------------
10   |      1 |      0 |XN Aggregate  (cost=6717.61..6  |
10   |      2 |      1 |  -> XN Merge Join DS_DIST_NO| Merge Cond:("outer"
10   |      3 |      2 |       -> XN Seq Scan on lis |
10   |      4 |      2 |       -> XN Seq Scan on sal |
(4 rows)
```

請考慮下列查詢：

```
select event.eventid, sum(pricepaid)
from event, sales
where event.eventid=sales.eventid
group by event.eventid order by 2 desc;

eventid |   sum
--------+----------
    289 | 51846.00
   7895 | 51049.00
   1602 | 50301.00
    851 | 49956.00
   7315 | 49823.00
...
```

如果此查詢的 ID 為 15，則下列系統資料表查詢會傳回已執行的計劃節點。在此情況下，會保留節點的順序，以顯示執行的實際順序：

```
select query,nodeid,parentid,substring(plannode from 1 for 56)
```

```
from svcs_explain where query=15 order by 1, 2 desc;

query|nodeid|parentid|                              substring
-----+------+--------+-------------------------------------------------------
15   |    8 |      7 |                                   -> XN Seq Scan on eve
15   |    7 |      5 |                              -> XN Hash(cost=87.98..87.9
15   |    6 |      5 |                              -> XN Seq Scan on sales(cos
15   |    5 |      4 |                         -> XN Hash Join DS_DIST_OUTER(cos
15   |    4 |      3 |                    -> XN HashAggregate(cost=862286577.07..
15   |    3 |      2 |           -> XN Sort(cost=1000862287175.47..10008622871
15   |    2 |      1 |  -> XN Network(cost=1000862287175.47..1000862287197.
15   |    1 |      0 |XN Merge(cost=1000862287175.47..1000862287197.46 rows=87
(8 rows)
```

下列查詢會擷取包含視窗函數之任何查詢計劃的查詢 ID：

```
select query, trim(plannode) from svcs_explain
where plannode like '%Window%';

query|                                      btrim
-----+-------------------------------------------------------------------------
26   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
27   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
(2 rows)
```

SVCS_PLAN_INFO

使用 SVCS_PLAN_INFO 資料表，以一組資料列查看查詢的 EXPLAIN 輸出。這是查看查詢計劃的其他方法。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 STL 的資料表，差別在於 STL 資料表僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_PLAN_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別符，其中節點會在查詢執行時映射至一個或多個步驟。
segment	integer	識別查詢區段的號碼。
step	integer	識別查詢步驟的號碼。
locus	integer	執行步驟的位置。若在運算節點上則為 0，若在領導者節點上則為 1。
plannode	integer	計劃節點的列舉值。請查看下表以取得 plannode 的列舉值。(SVCS_EXPLAIN 中的 PLANNODE 資料欄包含計劃節點文字。)
startupcost	double precision	傳回此步驟之第一列的預估相對成本。
totalcost	double precision	執行步驟的預估相對成本。
rows	bigint	步驟將產生之資料列的預估數目。
位元組	bigint	步驟將產生之位元組的預估數目。

範例查詢

下列範例會比較藉由使用 EXPLAIN 命令，以及藉由查詢 SVCS_PLAN_INFO 所傳回之簡單 SELECT 查詢的查詢計劃。

```
explain select * from category;
QUERY PLAN
------------------------------------------------------------
XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)
(1 row)
```

```
select * from category;
catid | catgroup | catname | catdesc
-------+----------+-----------+--------------------------------------------
1 | Sports | MLB | Major League Baseball
3 | Sports | NFL | National Football League
5 | Sports | MLS | Major League Soccer
...

select * from svcs_plan_info where query=256;

query | nodeid | segment | step | locus | plannode | startupcost | totalcost
| rows | bytes
-------+--------+---------+------+-------+----------+-------------+-----------+------
+-------
256 | 1 | 0 | 1 | 0 | 104 | 0 | 0.11 | 11 | 539
256 | 1 | 0 | 0 | 0 | 104 | 0 | 0.11 | 11 | 539
(2 rows)
```

在此範例中，PLANNODE 104 指的是 CATEGORY 資料表的循序掃描。

```
select distinct eventname from event order by 1;

eventname
----------------------------------------------------------------------------
.38 Special
3 Doors Down
70s Soul Jam
A Bronx Tale
...

explain select distinct eventname from event order by 1;

QUERY PLAN
------------------------------------------------------------------------------------------
XN Merge (cost=1000000000136.38..1000000000137.82 rows=576 width=17)
Merge Key: eventname
-> XN Network (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Send to leader
-> XN Sort (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
Sort Key: eventname
-> XN Unique (cost=0.00..109.98 rows=576 width=17)
```

```
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=17)
(8 rows)

select * from svcs_plan_info where query=240 order by nodeid desc;

query | nodeid | segment | step | locus | plannode | startupcost |
totalcost | rows | bytes
-------+--------+---------+------+-------+----------+-----------------
+------------------+------+--------
240 | 5 | 0 | 0 | 0 | 104 | 0                | 87.98   | 8798 | 149566
240 | 5 | 0 | 1 | 0 | 104 | 0                | 87.98   | 8798 | 149566
240 | 4 | 0 | 2 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 0 | 3 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 0 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 4 | 1 | 1 | 0 | 117 | 0                | 109.975 | 576  | 9792
240 | 3 | 1 | 2 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 3 | 2 | 0 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 2 | 2 | 1 | 0 | 123 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 1 | 3 | 0 | 0 | 122 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
(10 rows)
```

SVCS_QUERY_SUMMARY

您可以使用 SVCS_QUERY_SUMMARY 檢視來尋找查詢執行的一般相關資訊。

請留意，SVCS_QUERY_SUMMARY 中的資訊是從所有節點彙總而來。

> **Note**
>
> SVCS_QUERY_SUMMARY 檢視僅包含 Amazon Redshift 所完成查詢的相關資訊，不包含其他公用程式和 DDL 命令的相關資訊。如需 Amazon Redshift 所完成全部陳述式的完整清單和相關資訊 (包含 DDL 和公用程式命令)，則可查詢 SVL_STATEMENTTEXT 檢視。
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 SVL 的檢視，差別在於 SVL 檢視僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_QUERY_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需 SVL_QUERY_SUMMARY 的相關資訊，請參閱 SVL_QUERY_SUMMARY。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
stm	integer	串流：查詢的並行區段集。一個查詢擁有一或多個串流。
seg	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	執行的查詢步驟。
maxtime	bigint	此步驟執行的時間量上限 (微秒)。
avgtime	bigint	此步驟執行的平均時間 (微秒)。
rows	bigint	查詢步驟中包含的資料列數。
bytes	bigint	查詢步驟中包含的資料位元組數。
rate_row	double precision	每資料列的查詢執行率。
rate_byte	double precision	每位元組的查詢執行率。
label	text	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 scan tbl=100448 name =user)。三位數資料表 ID 通常是指暫時性資料表的掃描。當您看見 `tbl=0` 時，通常是指常數值的掃描。
is_diskbased	character(1)	此查詢步驟在叢集的任何節點上是否以磁碟型操作方式執行：true (**t**) 或 false (**f**)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。

欄名稱	資料類型	描述
is_rrscan	character(1)	若為 true (**t**)，表示已在步驟上使用範圍限制掃描。預設為 false (**f**)。
is_delayed_scan	character(1)	若為 true (**t**)，表示已在該步驟上使用延遲的掃描。預設為 false (**f**)。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之前已發出的資料列總數。

範例查詢

檢視查詢步驟的處理資訊

下列查詢顯示查詢 87 每個步驟的基本處理資訊：

```
select query, stm, seg, step, rows, bytes
from svcs_query_summary
where query = 87
order by query, seg, step;
```

此查詢會擷取查詢 87 的相關處理資訊，如下範例輸出所示：

```
 query | stm | seg | step |  rows  |  bytes
-------+-----+-----+------+--------+---------
87     |   0 |   0 |    0 |     90 |    1890
87     |   0 |   0 |    2 |     90 |     360
87     |   0 |   1 |    0 |     90 |     360
87     |   0 |   1 |    2 |     90 |    1440
87     |   1 |   2 |    0 | 210494 | 4209880
87     |   1 |   2 |    3 |  89500 |       0
87     |   1 |   2 |    6 |      4 |      96
87     |   2 |   3 |    0 |      4 |      96
87     |   2 |   3 |    1 |      4 |      96
87     |   2 |   4 |    0 |      4 |      96
87     |   2 |   4 |    1 |      1 |      24
87     |   3 |   5 |    0 |      1 |      24
87     |   3 |   5 |    4 |      0 |       0
(13 rows)
```

判斷查詢步驟是否溢出至磁碟

下列查詢顯示查詢 ID 為 1025 的任何查詢步驟 (請參閱 SVL_QLOG 檢視來了解如何取得查詢的查詢 ID) 是否溢出至磁碟或查詢完全在記憶體內執行：

```
select query, step, rows, workmem, label, is_diskbased
from svcs_query_summary
where query = 1025
order by workmem desc;
```

此查詢傳回下列範例輸出：

```
query| step|  rows   |  workmem    |  label         | is_diskbased
-----+-----+--------+-----------+---------------+--------------
1025 |  0  |16000000|  141557760 |scan tbl=9     | f
1025 |  2  |16000000|  135266304 |hash tbl=142   | t
1025 |  0  |16000000|  128974848 |scan tbl=116536| f
1025 |  2  |16000000|  122683392 |dist           | f
(4 rows)
```

透過掃描 IS_DISKBASED 的值，您可以檢視哪一個查詢步驟使用磁碟。針對查詢 1025，雜湊步驟在磁碟上執行。步驟可能在包含雜湊、彙總或排序步驟的磁碟上執行。若要只檢視磁碟型的查詢步驟，將 **`and is_diskbased = 't'`** 子句新增至以上範例的 SQL 陳述式。

SVCS_S3LIST

使用 SVCS_S3LIST 檢視以取得有關區段層級的 Amazon Redshift Spectrum 查詢詳細資訊。一個區段可以執行一個外部資料表掃描。此檢視衍生自 SVL_S3LIST 系統檢視，但不會顯示執行於並行擴展叢集之查詢的分割層級。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 SVL 的檢視，差別在於 SVL 檢視僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_S3LIST。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

如需 SVL_S3LIST 的相關資訊，請參閱 SVL_S3LIST。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。
segment	integer	區段號碼。一個查詢包含多個區段。
節點	integer	節點號碼。
eventtime	timestamp	記錄事件的時間 (以 UTC 表示)。
bucket	char(256)	Amazon S3 儲存貯體名稱。
字首	char(256)	Amazon S3 儲存貯體位置的字首。
recursive	char(1)	是否對子資料夾遞迴掃描。
retrieved_files	integer	所列檔案的數量。
max_file_size	bigint	所列檔案的檔案大小上限。
avg_file_size	double precision	所列檔案的平均檔案大小上限。
generated_splits	integer	檔案分割的數量。
avg_split_length	double precision	檔案分割的平均長度 (位元組)。
duration	bigint	檔案清單的期間 (微秒)。

範例查詢

以下範例會查詢前次查詢執行的 SVCS_S3LIST。

```
select *
```

```
from svcs_s3list
where query = pg_last_query_id()
order by query,segment;
```

SVCS_S3LOG

使用 SVCS_S3LOG 檢視以取得有關區段層級的 Redshift Spectrum 查詢的疑難排解詳細資訊。一個區段可以執行一個外部資料表掃描。此檢視衍生自 SVL_S3LOG 系統檢視，但不會顯示執行於並行擴展叢集之查詢的分割層級。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 SVL 的檢視，差別在於 SVL 檢視僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_S3LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

如需 SVL_S3LOG 的相關資訊，請參閱 SVL_S3LOG。

資料表欄

欄名稱	資料類型	描述
pid	integer	處理程序 ID。
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
節點	integer	節點號碼。
eventtime	timestamp	記錄事件的時間 (以 UTC 表示)。
message	char(512)	日誌項目訊息。

範例查詢

以下範例會查詢前次執行查詢的 SVCS_S3LOG。

```
select *
from svcs_s3log
where query = pg_last_query_id()
order by query,segment;
```

SVCS_S3PARTITION_SUMMARY

使用 SVCS_S3PARTITION_SUMMARY 檢視來取得區段層級的 Redshift Spectrum 查詢分割處理的摘要。一個區段可以執行一個外部資料表掃描。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 SVL 的檢視，差別在於 SVL 檢視僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_S3PARTITION_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

如需 SVL_S3PARTITION 的相關資訊，請參閱 SVL_S3PARTITION。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
segment	integer	區段號碼。一個查詢包含多個區段。
指派	char(1)	節點間分割指派的類型。
min_starttime	timestamp	分割區處理開始的時間 (以 UTC 表示)。
max_endtime	timestamp	分割區處理完成的時間 (以 UTC 表示)。

欄名稱	資料類型	描述
min_duration	bigint	節點針對此查詢所使用的最小分割處理時間 (微秒)。
max_duration	bigint	節點針對此查詢所使用的最大分割處理時間 (微秒)。
avg_duration	bigint	節點針對此查詢所使用的平均分割處理時間 (微秒)。
total_partitions	integer	外部資料表中分割區的總數。
qualified_partitions	integer	合格分割區總數。
min_assigned_partitions	integer	在一個節點上指派的最小分割區數量。
max_assigned_partitions	integer	在一個節點上指派的最大分割區數量。
avg_assigned_partitions	bigint	在一個節點上指派的平均分割區數量。

範例查詢

以下範例會取得前次執行查詢的分割區掃描詳細資訊。

```
select query, segment, assignment, min_starttime, max_endtime, min_duration,
 avg_duration
from svcs_s3partition_summary
where query = pg_last_query_id()
order by query,segment;
```

SVCS_S3QUERY_SUMMARY

使用 SVCS_S3QUERY_SUMMARY 檢視以取得在系統上執行之所有 Redshift Spectrum 查詢 (S3 查詢) 摘要。一個區段可以執行一個外部資料表掃描。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 SVL 的檢視，差別在於 SVL 檢視僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_S3QUERY_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

如需 SVL_S3QUERY 的相關資訊，請參閱 SVL_S3QUERY。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行給定項目的使用者 ID。
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
starttime	timestamp	此區段中的 Redshift Spectrum 查詢開始執行的時間 (以 UTC 表示)。一個區段可以有一個外部資料表掃描。
endtime	timestamp	此區段中的 Redshift Spectrum 查詢完成的時間 (以 UTC 表示)。一個區段可以有一個外部資料表掃描。

欄名稱	資料類型	描述
elapsed	integer	此區段中的 Redshift Spectrum 查詢執行所需的時間長度 (微秒)。
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 **1**。如果查詢執行至完成，欄位會包含 **0**。
external_table_name	char(136)	資料表外部名稱之名稱的內部格式，適用於外部資料表掃描。
file_format	character(16)	外部資料表資料的檔案格式。
is_partitioned	char(1)	若為 true (**t**)，此欄位值表示外部資料表已遭分割。
is_rrscan	char(1)	若為 true (**t**)，此欄位值表示已套用範圍限定的掃描。
is_nested	varchar(1)	若為 true (**t**)，此欄位值表示已存取巢狀資料欄資料類型。
s3_scanned_rows	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的列數。
s3_scanned_bytes	bigint	根據壓縮檔案，從 Amazon S3 經掃描並送至 Redshift Spectrum 層的位元組數。
s3query_returned_rows	bigint	從 Redshift Spectrum 層傳回至叢集的資料列數。
s3query_returned_bytes	bigint	從 Redshift Spectrum 層傳回至叢集的位元組數。傳回至 Amazon Redshift 的大量資料可能會影響系統效能。
files	integer	此 Redshift Spectrum 查詢已處理的檔案數。限制平行處理之好處的少數檔案。
files_max	integer	在某分割上處理的檔案數上限。

欄名稱	資料類型	描述
files_avg	integer	在某分割上處理的平均檔案數。
splits	bigint	為此區段處理的分割數。在此分割上處理的分割數。有了大型可分割資料檔案 (例如，大於 512 MB 的資料檔案)，Redshift Spectrum 會嘗試將檔案分割至多個 S3 請求來進行平行處理。
splits_max	integer	在此分割上處理的分割數上限。
splits_avg	bigint	在此分割上處理的平均分割數。
total_split_size	bigint	處理的所有分割大小總計。
max_split_size	bigint	處理的分割大小上限 (位元組)。
avg_split_size	bigint	處理的平均分割大小 (位元組)。
total_retries	bigint	此區段中 Redshift Spectrum 查詢的重試數總計。
max_retries	integer	一個個別處理檔案的重試數上限。
max_request_duration	bigint	個別檔案請求的最大持續期間 (微秒)。長時間執行的查詢可能表示存在瓶頸。
avg_request_duration	bigint	檔案請求的平均持續時間 (微秒)。
max_request_parallelism	integer	此 Redshift Spectrum 查詢在某個分割上的最大平行請求數。

欄名稱	資料類型	描述
avg_request_parallelism	double precision	此 Redshift Spectrum 查詢在某個分割上的平均平行請求數。
total_slowdown_count	bigint	外部資料表掃描期間發生，具有降速錯誤的 Amazon S3 請求總數。
max_slowdown_count	integer	在某分割上外部資料表掃描期間發生，具有降速錯誤的最大 Amazon S3 請求數。

範例查詢

以下範例會取得前次執行查詢的掃描步驟詳細資訊。

```
select query, segment, elapsed, s3_scanned_rows, s3_scanned_bytes,
 s3query_returned_rows, s3query_returned_bytes, files
from svcs_s3query_summary
where query = pg_last_query_id()
order by query,segment;
```

```
query | segment | elapsed | s3_scanned_rows | s3_scanned_bytes | s3query_returned_rows
 | s3query_returned_bytes | files
------+---------+---------+-----------------+------------------+----------------------
+------------------------+------
 4587 |       2 |   67811 |               0 |                0 |                    0
 |                      0 |     0
 4587 |       2 |  591568 |          172462 |         11260097 |                 8513
 |                 170260 |     1
 4587 |       2 |  216849 |               0 |                0 |                    0
 |                      0 |     0
 4587 |       2 |  216671 |               0 |                0 |                    0
 |                      0 |     0
```

SVCS_STREAM_SEGS

列出串流與並行區段之間的關係。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 STL 的資料表，差別在於 STL 資料表僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_STREAM_SEGS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
串流	integer	查詢的並行區段集。
segment	integer	識別查詢區段的號碼。

範例查詢

若要針對最新查詢檢視串流與並行區段之間的關係，請輸入下列查詢：

```
select *
from svcs_stream_segs
where query = pg_last_query_id();

 query | stream | segment
-------+--------+---------
    10 |      1 |       2
    10 |      0 |       0
    10 |      2 |       4
    10 |      1 |       3
    10 |      0 |       1
(5 rows)
```

SVCS_UNLOAD_LOG

使用 SVCS_UNLOAD_LOG 來取得 UNLOAD 操作的詳細資訊。

SVCS_UNLOAD_LOG 會針對 UNLOAD 陳述式所建立的每一個檔案記錄一個資料列。例如，若 UNLOAD 建立 12 個檔案，則 SVCS_UNLOAD_LOG 包含 12 個對應資料列。此檢視衍生自 STL_UNLOAD_LOG 系統資料表，但不會顯示執行於並行擴展叢集之查詢的分割層級。

> **Note**
>
> 字首為 SVCS 的系統檢視可提供查詢的詳細資訊，包括主要叢集與並行擴展叢集上的查詢。這些檢視類似字首為 STL 的資料表，差別在於 STL 資料表僅提供執行於主要叢集之查詢的資訊。

所有使用者都可看見 SVCS_UNLOAD_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。
pid	integer	與查詢陳述式相關聯的處理程序 ID。
路徑	character(1280)	檔案的完整 Amazon S3 物件路徑。
start_time	timestamp	UNLOAD 作業的開始時間。
end_time	timestamp	UNLOAD 作業的結束時間。
line_count	bigint	已卸載至檔案的行數 (列數)。
transfer_size	bigint	已傳輸的位元組數目。
file_format	character(10)	已卸載檔案的格式。

範例查詢

若要取得已由 UNLOAD 命令寫入至 Amazon S3 的檔案清單，您可以在 UNLOAD 完成之後呼叫 Amazon S3 清單操作；不過，根據您發出呼叫的速度，清單可能不完整，因為 Amazon S3 清單操作最終會保持一致。若要立即取得完整、授權的清單，請查詢 SVCS_UNLOAD_LOG。

下列查詢會傳回已由 UNLOAD 針對前一個執行的查詢建立之檔案的路徑名稱：

```
select query, substring(path,0,40) as path
from svcs_unload_log
where query = pg_last_query_id()
order by path;
```

此命令會傳回下列範例輸出：

```
 query |             path
 ------+---------------------------------
  2320 | s3://my-bucket/venue0000_part_00
  2320 | s3://my-bucket/venue0001_part_00
  2320 | s3://my-bucket/venue0002_part_00
  2320 | s3://my-bucket/venue0003_part_00
(4 rows)
```

主要叢集的 SVL 檢視

SVL 檢視是 Amazon Redshift 中包含 STL 資料表和日誌之參考的系統檢視，可了解詳細資訊。

這些檢視可讓您快速輕鬆存取在那些資料表中找到的常見查詢資料。

> **Note**
>
> SVL_QUERY_SUMMARY 檢視僅包含 Amazon Redshift 執行之查詢的相關資訊，不包含其他公用程式和 DDL 命令的相關資訊。如需 Amazon Redshift 執行的所有陳述式 (包括 DDL 和公用程式命令) 的完整清單和相關資訊，您可以查詢 SVL_STATEMENTTEXT 檢視。

主題

- SVL_AUTO_WORKER_ACTION
- SVL_COMPILE

- SVL_DATASHARE_CHANGE_LOG
- SVL_DATASHARE_CROSS_REGION_USAGE
- SVL_DATASHARE_USAGE_CONSUMER
- SVL_DATASHARE_USAGE_PRODUCER
- SVL_FEDERATED_QUERY
- SVL_MULTI_STATEMENT_VIOLATIONS
- SVL_MV_REFRESH_STATUS
- SVL_QERROR
- SVL_QLOG
- SVL_QUERY_METRICS
- SVL_QUERY_METRICS_SUMMARY
- SVL_QUERY_QUEUE_INFO
- SVL_QUERY_REPORT
- SVL_QUERY_SUMMARY
- SVL 還原 _ 更改表 _ 進度
- SVL_S3LIST
- SVL_S3LOG
- SVL_S3PARTITION
- SVL_S3PARTITION_SUMMARY
- SVL_S3QUERY
- SVL_S3QUERY_SUMMARY
- SVL_S3RETRIES
- SVL_SPATIAL_SIMPLIFY
- SVL_SPECTRUM_SCAN_ERROR
- SVL_STATEMENTTEXT
- SVL_STORED_PROC_CALL
- SVL_STORED_PROC_MESSAGES
- SVL_TERMINATE

- SVL_UDF_LOG
- SVL_USER_INFO
- SVL_VACUUM_PERCENTAGE

SVL_AUTO_WORKER_ACTION

記錄 Amazon Redshift 對自動最佳化而定義的資料表執行的自動操作。

只有超級使用者可以看到 SVL_AUTO_WORKER_ACTION。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
table_id	integer	資料表識別碼。
type	character (32)	建議的類型。可能的值是 distkey 和 sortkey。
status	character (128)	建議的完成狀態。可能的值為 Start、Complete、Skipped、Abort、Checkpoint 和 Failed
eventtime	timestamp	`status` 欄的時間戳記。
sequence	integer	截斷 `previous_state` 值的序列號。當單一 `previous_state` 包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
previous_state	character (200)	套用建議之前，資料表的先前分散樣式和排序索引鍵。該值會被截斷為 200 個字元的增量。

`status` 欄的值的一些飯粒如下：

- 略過：找不到資料表。
- 略過：建議是空的。

- 略過：套用排序索引鍵建議已停用。
- 略過：重試超過資料表的上限。
- 略過：資料表欄已變更。
- 中止：此資料表不是 AUTO。
- 中止：最近已轉換此資料表。
- 中止：此資料表超過資料表大小臨界值。
- 中止：此資料表已經是建議的樣式。
- 檢查點：進度為 *21.9963%*。

範例查詢

在下列範例中，結果中的列顯示 Amazon Redshift 所採取的動作。

```
select table_id, type, status, eventtime, sequence, previous_state
from SVL_AUTO_WORKER_ACTION;
```

```
 table_id |  type   |                         status                         |
 eventtime          | sequence | previous_state
----------+---------+--------------------------------------------------------
+----------------------------+----------+----------------
   118082 | sortkey | Start                                                  | 2020-08-22
 19:42:20.727049 | 0        |
   118078 | sortkey | Start                                                  | 2020-08-22
 19:43:54.728819 | 0        |
   118082 | sortkey | Start                                                  | 2020-08-22
 19:42:52.690264 | 0        |
   118072 | sortkey | Start                                                  | 2020-08-22
 19:44:14.793572 | 0        |
   118082 | sortkey | Failed                                                 | 2020-08-22
 19:42:20.728917 | 0        |
   118078 | sortkey | Complete                                               | 2020-08-22
 19:43:54.792705 | 0        | SORTKEY: None;
   118086 | sortkey | Complete                                               | 2020-08-22
 19:42:00.72635  | 0        | SORTKEY: None;
   118082 | sortkey | Complete                                               | 2020-08-22
 19:43:34.728144 | 0        | SORTKEY: None;
   118072 | sortkey | Skipped:Retry exceeds the maximum limit for a table. | 2020-08-22
 19:44:46.706155 | 0        |
```

```
  118086 | sortkey | Start                                                | 2020-08-22
19:42:00.685255 | 0       |
  118082 | sortkey | Start                                                | 2020-08-22
19:43:34.69531  | 0       |
  118072 | sortkey | Start                                                | 2020-08-22
19:44:46.703331 | 0       |
  118082 | sortkey | Checkpoint: progress 14.755079%                      | 2020-08-22
19:42:52.692828 | 0       |
  118072 | sortkey | Failed                                               | 2020-08-22
19:44:14.796071 | 0       |
  116723 | sortkey | Abort:This table is not AUTO.                        | 2020-10-28
05:12:58.479233 | 0       |
  110203 | distkey | Abort:This table is not AUTO.                        | 2020-10-28
05:45:54.67259  | 0       |
```

SVL_COMPILE

記錄查詢之每個查詢區段的編譯時間和位置。

所有使用者都可看見 SVL_COMPILE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> SVL_COMPILE 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_QUERY_HISTORY。SYS 監視檢視中的資料會格式化為更易於使用和理解。

如需 SVCS_COMPILE 的相關資訊，請參閱 SVCS_COMPILE。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
segment	integer	要編譯的查詢區段。
locus	integer	執行區段的位置。**1** (若在運算節點上) 和 **2** (若在領導節點上)。
starttime	timestamp	編譯開始的時間，以 UTC 表示。
endtime	timestamp	編譯結束的時間，以 UTC 表示。
compile	integer	**0** 表示編譯是否重新使用，**1** 表示區段是否編譯。

範例查詢

在此範例中，查詢 35878 和 35879 執行相同的 SQL 陳述式。查詢 35878 的編譯欄位針對四個查詢區段顯示 1，此值表示該區段已編譯。查詢 35879 在每個區段的編譯欄位中顯示 0，表示該區段不需要重新編譯。

```
select userid, xid,  pid, query, segment, locus,
datediff(ms, starttime, endtime) as duration, compile
from svl_compile
where query = 35878 or query = 35879
order by query, segment;

 userid |  xid   |  pid  | query | segment | locus | duration | compile
--------+--------+-------+-------+---------+-------+----------+---------
    100 | 112780 | 23028 | 35878 |       0 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       1 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       2 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       3 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       4 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       5 |     1 |        0 |       0
    100 | 112780 | 23028 | 35878 |       6 |     1 |     1380 |       1
    100 | 112780 | 23028 | 35878 |       7 |     1 |     1085 |       1
    100 | 112780 | 23028 | 35878 |       8 |     1 |     1197 |       1
    100 | 112780 | 23028 | 35878 |       9 |     2 |      905 |       1
    100 | 112782 | 23028 | 35879 |       0 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       1 |     1 |        0 |       0
```

```
    100 | 112782 | 23028 | 35879 |       2 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       3 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       4 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       5 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       6 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       7 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       8 |     1 |        0 |       0
    100 | 112782 | 23028 | 35879 |       9 |     2 |        0 |       0
(20 rows)
```

SVL_DATASHARE_CHANGE_LOG

記錄用於追蹤生產者和消費者叢集上資料共用變更的合併檢視。

所有使用者都可看見 SVL_DATASHARE_CHANGE_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_DATASHARE_CHANGE_LOG 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	採取動作之使用者的 ID。
使用者名稱	varchar(128)	採取動作的使用者名稱。
pid	integer	程序的 ID。
xid	bigint	交易的 ID。
share_id	integer	受影響資料共用的 ID。
share_name	varchar(128)	資料共用的名稱。
source_database_id	integer	資料共用所屬資料庫的 ID。

欄名稱	資料類型	描述
source_database_name	varchar(128)	資料共用所屬資料庫的名稱。
consumer_database_id	integer	從資料共用匯入之資料庫的 ID。
consumer_database_name	varchar(128)	從資料共用匯入之資料庫的名稱。
arn	varchar(192)	支援匯入資料庫之資源的 ARN。
recordtime	timestamp	動作的時間戳記。
動作	varchar(128)	正在執行的動作。可能的值包括創建數據保護，刪除數據清理，授予改變，撤銷改變，授予共享，撤銷份額，更改添加，更改刪除，更改集合，授予使用情況，撤銷使用情況，創建數據庫，授予或撤銷共享數據庫的使用量，刪除共享數據庫，改變共享數據庫。
status	integer	動作的狀態。可能的值為 SUCCESS 和 ERROR-ERROR CODE。
share_object_type	varchar(64)	新增至資料共用或從資料共用中移除之資料庫物件的類型。可能的值包括 schemas、tables、columns、functions 和 views。這是生產者叢集的欄位。
share_object_id	integer	新增至資料共用或從資料共用中移除之資料庫物件的 ID。這是生產者叢集的欄位。
share_object_name	varchar(128)	新增至資料共用或從資料共用中移除之資料庫物件的名稱。這是生產者叢集的欄位。
target_user_type	varchar(16)	授與權限的使用者或群組類型。這是生產者和消費者叢集的欄位。

欄名稱	資料類型	描述
target_userid	integer	授與權限的使用者或群組 ID。這是生產者和消費者叢集的欄位。
target_username	varchar(128)	授與權限的使用者或群組名稱。這是生產者和消費者叢集的欄位。
consumer_account	varchar(16)	資料消費者的帳戶 ID。這是生產者叢集的欄位。
consumer_namespace	varchar(64)	資料消費者帳戶的命名空間。這是生產者叢集的欄位。
producer_account	varchar(16)	資料共用所屬生產者帳戶的帳戶 ID。這是消費者叢集的欄位。
producer_namespace	varchar(64)	資料共用所屬產品帳戶的命名空間。這是消費者叢集的欄位。
attribute_name	varchar(64)	資料共用或共用資料庫的屬性名稱。
attribute_value	varchar(128)	資料共用或共用資料庫的屬性值。
message	varchar(512)	動作失敗時的錯誤訊息。

範例查詢

下列範例顯示 SVL_DATASHARE_CHANGE_LOG 檢視。

```
SELECT DISTINCT action
FROM svl_datashare_change_log
WHERE share_object_name LIKE 'tickit%';

         action
 ----------------------
```

```
  "ALTER DATASHARE ADD"
```

SVL_DATASHARE_CROSS_REGION_USAGE

使用 SVL_DATASHARE_CROSS_REGION_USAGE 檢視，可取得跨區域資料共用查詢所造成之跨區域資料傳輸使用量的摘要。SVL_DATASHARE_CROSS_REGION_USAGE 彙總區段層級的詳細資料。

所有使用者都可看見 SVL_DATASHARE_CROSS_REGION_USAGE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_DATASHARE_CROSS_REGION_USAGE 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢的 ID。使用此值來聯結各種其他系統資料表與檢視。
segment	bigint	區段的號碼。查詢包含多個區段，每個區段包含一或多個步驟。
start_time	time	資料傳輸開始的時間 (以 UTC 為單位)。
end_time	time	資料傳輸結束的時間 (以 UTC 為單位)。
transferred_data	bigint	從生產者區域傳輸至消費者區域的資料位元組數量。
source_region	char(25)	查詢傳輸資料的來源生產者區域。
recordtime	timestamp	記錄動作的時間。

範例查詢

下列範例顯示 SVL_DATASHARE_CROSS_REGION_USAGE 檢視。

```
SELECT query, segment, transferred_data, source_region
from svl_datashare_cross_region_usage
where query = pg_last_query_id()
order by query,segment;

  query | segment | transferred_data | source_region
--------+---------+------------------+---------------
 200048 |       2 |          4194304 |    us-west-1
 200048 |       2 |          4194304 |    us-east-2
```

SVL_DATASHARE_USAGE_CONSUMER

記錄資料共用的活動和使用情況。此檢視僅與消費者叢集相關。

所有使用者都可看見 SVL_DATASHARE_USAGE_CONSUMER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_DATASHARE_USAGE_CONSUMER 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	發出要求的使用者 ID。
pid	integer	執行查詢之領導者程序的 ID。
xid	bigint	目前交易的內容。
request_id	varchar(50)	要求之 API 呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。
transaction_uid	varchar(50)	交易的唯一 ID。
recordtime	timestamp	記錄動作的時間。

欄名稱	資料類型	描述
status	integer	要求的 API 呼叫的狀態。
error	varchar(512)	錯誤的訊息。

範例查詢

下列範例顯示 SVL_DATASHARE_USAGE_CONSUMER 檢視。

```
SELECT request_type, status, trim(error) AS error
FROM svl_datashare_usage_consumer

  request_type  | status | error
----------------+--------+--------
 "GET RELATION" |   0    |
```

SVL_DATASHARE_USAGE_PRODUCER

記錄資料共用的活動和使用情況。此檢視僅與生產者叢集相關。

所有使用者都可看見 SVL_DATASHARE_USAGE_PRODUCER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_DATASHARE_USAGE_PRODUCER 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
share_id	integer	資料共用的物件 ID (OID)。
share_name	varchar(128)	資料共用的名稱。

欄名稱	資料類型	描述
request_id	varchar(50)	要求之 API 呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。
object_type	varchar(64)	從資料共用共用的物件類型。可能的值包括 schemas、tables、columns、functions 和 views。
object_oid	integer	從資料共用共用的物件 ID。
object_name	varchar(128)	要從資料共用共用的物件名稱。
consumer_account	varchar(16)	資料共用共用到的消費者帳戶的帳戶。
consumer_namespace	varchar(64)	資料共用共用到的使用者帳戶的命名空間。
consumer_transaction_uid	varchar(50)	消費者叢集上陳述式的唯一交易 ID。
recordtime	timestamp	記錄動作的時間。
status	integer	資料共用的狀態。
error	varchar(512)	錯誤的訊息。
consumer_region	char(64)	取用者叢集所在的區域。

範例查詢

下列範例顯示 SVL_DATASHARE_USAGE_PRODUCER 檢視。

```
SELECT DISTINCT request_type
FROM svl_datashare_usage_producer
WHERE object_name LIKE 'tickit%';

   request_type
 ------------------
   "GET RELATION"
```

SVL_FEDERATED_QUERY

使用 SVL_FEDERATED_QUERY 檢視，可檢視有關聯合查詢呼叫的資訊。

所有使用者都可看見 SVL_FEDERATED_QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_EXTERNAL_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行查詢的使用者 ID。
xid	bigint	交易 ID。
pid	integer	執行查詢之領導者程序的 ID。
query	integer	聯合呼叫的查詢 ID。
sourcetype	character (32)	聯合呼叫來源類型，例如 "PG"。
recordtime	timestamp	傳送查詢以進行聯合的時間。使用 UTC。
querytext	character (4000)	傳送到遠端 PostgreSQL 引擎以執行的查詢字串。
num_rows	bigint	聯合查詢傳回的資料列數。

欄名稱	資料類型	描述
num_bytes	bigint	聯合查詢所傳回的位元組數量。
duration	bigint	從游標呼叫擷取資料列所花費的時間 (微秒)。這是執行聯合查詢所花費的時間，以及獲取結果。

範例查詢

若要顯示聯合查詢呼叫的相關資訊，請執行下列查詢。

```
select query, trim(sourcetype) as type, recordtime, trim(querytext) as "PG Subquery"
 from svl_federated_query where query = 4292;

 query | type |         recordtime         |                          pg subquery
-------+------+----------------------------
+-------------------------------------------------------------
  4292 | PG   | 2020-03-27 04:29:58.485126 | SELECT "level" FROM functional.employees
 WHERE ("level" >= 6)
(1 row)
```

SVL_MULTI_STATEMENT_VIOLATIONS

使用 SVL_MULTI_STATMENT_VIOLATIONS 檢視來取得系統上執行的所有違反交易區塊限制的 SQL 命令的完整記錄。

當您執行 Amazon Redshift 限制在交易區塊或多重陳述式請求內的下列任何 SQL 命令時，就會發生違規：

- CREATE DATABASE
- DROP DATABASE
- ALTER TABLE APPEND
- CREATE EXTERNAL TABLE
- DROP EXTERNAL TABLE
- RENAME EXTERNAL TABLE

- ALTER EXTERNAL TABLE
- CREATE TABLESPACE
- DROP TABLESPACE
- CREATE LIBRARY
- DROP LIBRARY
- REBUILDCAT
- INDEXCAT
- REINDEX DATABASE
- VACUUM
- GRANT
- COPY

> **Note**
>
> 如果此檢視中有任何項目，請變更對應的應用程式和 SQL 命令檔。我們建議您變更應用程式程式碼，將這些受限制 SQL 命令的使用移至交易區塊之外。如果您需要進一步協助，請聯絡 Sup AWS port 部門。

所有使用者都可看見 SVL_MULTI_STATEMENT_VIOLATIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	造成違規的使用者 ID。
database	character(32)	使用者連接的資料庫名稱。
cmdname	character(20)	無法在交易區塊或多重陳述式要求內執行的命令名稱。例如，CREATE DATABASE、DROP DATABASE、ALTER TABLE APPEND、CREATE EXTERNAL TABLE、DROP

欄名稱	資料類型	描述
		EXTERNAL TABLE、RENAME EXTERNAL TABLE、ALTER EXTERNAL TABLE、CREATE LIBRARY、DROP LIBRARY、REBUILDCAT、INDEXCAT、REINDEX DATABASE、VACUUM、外部資源上的 GRANT、CLUSTER、COPY、CREATE TABLESPACE 以及 DROP TABLESPACE。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	陳述式的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位為空白。
starttime	timestamp	陳述式開始執行的確切時間，精確度為六位數的小數秒，例如：**2009-06-12 11:29:19.131358**
endtime	timestamp	陳述式完成執行的確切時間，精確度為六位數的小數秒，例如：**2009-06-12 11:29:19.193640**
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
type	varchar(10)	SQL 陳述式的類型：**QUERY**、**DDL** 或 **UTILITY**。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回多個有違規的陳述式。

```
select * from svl_multi_statement_violations order by starttime asc;

userid | database | cmdname | xid | pid | label | starttime | endtime | sequence | type
 | text
```

```
================================================================================================
1 | dev | CREATE DATABASE | 1034 | 5729 |label1 | ******** | ******* | 0 | DDL |
 create table c(b int);
1 | dev | CREATE DATABASE | 1034 | 5729 |label1 | ******** | ******* | 0 | UTILITY |
 create database b;
1 | dev | CREATE DATABASE | 1034 | 5729 |label1 | ******** | ******* | 0 | UTILITY |
 COMMIT
...
```

SVL_MV_REFRESH_STATUS

針對具體化視觀表的重新整理活動，SVL_MV_REFRESH_STATUS 檢視會包含一個資料列。

如需具體化視觀表的相關資訊，請參閱 在 Amazon Redshift 中建立具體化視觀表。

所有使用者都可看見 SVL_MV_REFRESH_STATUS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_MV_REFRESH_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
db_name	char(128)	包含具體化視觀表的資料庫。
userid	bigint	執行重新整理的使用者 ID。
schema_name	char(128)	具體化視觀表的結構描述。
mv_name	char(128)	具體化視觀表名稱。
xid	bigint	重新整理的交易 ID。
starttime	timestamp	重新整理的開始時間。
endtime	timestamp	重新整理的結束時間。
status	text	重新整理的狀態。範例值包括下列各項： • 重新整理以累加方式成功更新 MV

欄名稱	資料類型	描述
		如果它是用於串流的具體化視觀表，則訊息可能會有關於記錄數目的其他限定詞。這些索引標籤包括以下項目： • 串流未傳回任何新資料 - 未擷取任何記錄。 • 從串流收到的所有記錄被略過 - 已擷取記錄，但由於錯誤，所有記錄都已略過。 • 一些串流記錄被略過-記錄被檢索，但由於錯誤，有些被略過。 如果沒有限定元，則至少會擷取一筆記錄，且具體化視觀表中的所有記錄都可用。還有一個可能的限定詞： • 串流可能包含更多資料 - 在 Amazon Redshift 判斷沒有進一步的記錄可供使用之前，重新整理即結束。該串流可能是最新的，但尚未得到 Amazon Redshift 的確認。 • 重新整理成功從頭重新運算 MV • 重新整理以累加方式將 MV 部分更新至作用中交易 • 已經更新 MV • 重新整理失敗。已重新命名基底資料表資料欄 • 重新整理失敗。已變更基底資料表資料欄類型 • 重新整理失敗。已重新命名基底資料表 • 內部錯誤導致的重新整理失敗。 • 重新整理失敗。已捨棄基底資料表資料欄 • 重新整理失敗。已重新命名 MV 的結構描述 • 重新整理失敗。找不到 MV • 自動重新整理因使用者工作負載過多而中止 • 重新整理失敗。可序列化隔離違規
refresh_type	char(32)	重新整理類型的定義。範例值包括手動和自動。

範例查詢

若要檢視具體化視觀表的重新整理狀態，請執行下列查詢。

```
select * from svl_mv_refresh_status;
```

此查詢傳回下列範例輸出：

```
 db_name | userid |  schema   |  name   |  xid  |         starttime          |
       endtime           |                        status                        |
 refresh_type
---------+--------+-----------+---------+-------+----------------------------
+----------------------------+------------------------------------------------------
+-----------------
 dev     |    169 | mv_schema | mv_test |  6640 | 2020-02-14 02:26:53.497935 |
 2020-02-14 02:26:53.556156 | Refresh successfully recomputed MV from scratch     |
 Manual
 dev     |    166 | mv_schema | mv_test |  6517 | 2020-02-14 02:26:39.287438 |
 2020-02-14 02:26:39.349539 | Refresh successfully updated MV incrementally       |
 Auto
 dev     |    162 | mv_schema | mv_test |  6388 | 2020-02-14 02:26:27.863426 |
 2020-02-14 02:26:27.918307 | Refresh successfully recomputed MV from scratch     |
 Manual
 dev     |    161 | mv_schema | mv_test |  6323 | 2020-02-14 02:26:20.020717 |
 2020-02-14 02:26:20.080002 | Refresh successfully updated MV incrementally       |
 Auto
 dev     |    161 | mv_schema | mv_test |  6301 | 2020-02-14 02:26:05.796146 |
 2020-02-14 02:26:07.853986 | Refresh successfully recomputed MV from scratch     |
 Manual
 dev     |    153 | mv_schema | mv_test |  6024 | 2020-02-14 02:25:18.762335 |
 2020-02-14 02:25:20.043462 | MV was already updated                              |
 Manual
 dev     |    143 | mv_schema | mv_test |  5557 | 2020-02-14 02:24:23.100601 |
 2020-02-14 02:24:23.100633 | MV was already updated                              |
 Manual
 dev     |    141 | mv_schema | mv_test |  5447 | 2020-02-14 02:23:54.102837 |
 2020-02-14 02:24:00.310166 | Refresh successfully updated MV incrementally       |
 Auto
 dev     |      1 | mv_schema | mv_test |  5329 | 2020-02-14 02:22:26.328481 |
 2020-02-14 02:22:28.369217 | Refresh successfully recomputed MV from scratch     |
 Auto
```

```
dev     |    138 | mv_schema | mv_test |  5290 | 2020-02-14 02:21:56.885093 |
2020-02-14 02:21:56.885098 | Refresh failed. MV was not found                    |
Manual
```

SVL_QERROR

SVL_QERROR 檢視已棄用。

SVL_QLOG

SVL_QLOG 檢視包含針對資料庫執行之所有查詢的記錄。

Amazon Redshift 會從 STL_QUERY 資料表以可讀資訊子集的形式建立 SVL_QLOG 檢視。使用此資料表來找出最近執行之查詢的查詢 ID，或查看完成查詢需要多長時間。

所有使用者都可看見 SVL_QLOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
xid	bigint	交易 ID。
pid	integer	與查詢相關聯的處理程序 ID。
starttime	timestamp	陳述式開始執行的確切時間，精確度為六位數的小數秒，例如：**2009-06-12 11:29:19.131358**
endtime	timestamp	陳述式完成執行的確切時間，精確度為六位數的小數秒，例如：**2009-06-12 11:29:19.193640**

欄名稱	資料類型	描述
elapsed	bigint	查詢執行的所需時間 (微秒)。
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 **1**。如果查詢執行至完成，欄位會包含 **0**。因工作負載管理目的而取消且後續會再重新啟動的查詢也會在此欄位中擁有值 **1**。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位值為 `default`。
substring	character(60)	遭截斷的查詢文字。
source_query	integer	如果查詢使用了結果快取，則為快取結果來源之查詢的查詢 ID。如果未使用結果快取，此欄位的值為 NULL。
concurrency_scaling_status_txt	text	查詢是執行於主要叢集或並行擴展叢集的描述。
from_sp_call	integer	如果是從預存程序呼叫查詢，則為程序呼叫的查詢 ID。如果查詢不是在預存程序中執行，則此欄位為 NULL。

範例查詢

下列範例傳回使用者 `userid = 100` 執行之最近五個資料庫查詢的查詢 ID、執行時間和遭截斷的查詢文字。

```
select query, pid, elapsed, substring from svl_qlog
where userid = 100
order by starttime desc
limit 5;

 query  |  pid  | elapsed  |          substring
--------+-------+----------+----------------------------------------------
 187752 | 18921 | 18465685 | select query, elapsed, substring from svl_...
 204168 |  5117 |    59603 | insert into testtable values (100);
```

```
 187561 | 17046 |  1003052 | select * from pg_table_def where tablename...
 187549 | 17046 |  1108584 | select * from STV_WLM_SERVICE_CLASS_CONFIG
 187468 | 17046 |  5670661 | select * from pg_table_def where schemaname...
(5 rows)
```

下列範例會傳回 SQL 指令碼名稱 (LABEL 欄位) 和已取消查詢的歷經時間 (**aborted=1**)：

```
select query, elapsed, trim(label) querylabel
from svl_qlog where aborted=1;

 query | elapsed  |       querylabel
-------+----------+-------------------------
    16 |  6935292 | alltickittablesjoin.sql
(1 row)
```

SVL_QUERY_METRICS

SVL_QUERY_METRICS 檢視顯示已完成之查詢的指標。此檢視衍生自 STL_QUERY_METRICS 系統資料表。請使用此檢視中的值來協助決定臨界值，以定義查詢監控規則。如需詳細資訊，請參閱 WLM 查詢監控規則。

所有使用者都可看見 SVL_QUERY_METRICS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
service_class	integer	WLM 查詢佇列 ID (服務類別)。查詢佇列定義於 WLM 組態。只會回報使用者定義之佇列的指標。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
dimension	varchar(24)	指標回報的維度。可能的值為查詢、區段和步驟。

欄名稱	資料類型	描述
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。如果區段值為 0，指標區段值將累計至查詢層級。
step	integer	所執行步驟類型的 ID。步驟類型的說明會在 `step_label` 欄位中顯示。
step_label	varchar(30)	執行的步驟類型。
query_cpu_time	bigint	查詢所用的 CPU 時間 (秒)。CPU 時間與查詢執行時間不同。
query_blocks_read	bigint	查詢所讀取的 1 MB 區塊數。
query_execution_time	bigint	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。如需排入佇列的時間，請參閱 `query_queue_time` 。
query_cpu_usage_percent	bigint	查詢所用的 CPU 容量百分比。
query_temp_blocks_to_disk	bigint	查詢用來寫入中繼結果之磁碟空間量 (MB)。
segment_execution_time	bigint	單一區段經過的執行時間 (秒)。
cpu_skew	numeric(38,2)	任何分割 CPU 用量與所有分割之平均 CPU 用量的比率。此指標在區段層級中定義。
io_skew	numeric(38,2)	任何分割讀取區塊數上限 (I/O) 與所有分割平均讀取區塊數的比率。
scan_row_count	bigint	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。
join_row_count	bigint	聯結步驟中處理的資料列數。

欄名稱	資料類型	描述
nested_loop_join_row_count	bigint	巢狀迴圈聯結中的資料列數。
return_row_count	bigint	查詢傳回的資料列數。
spectrum_scan_row_count	bigint	Amazon S3 中的 Amazon Redshift Spectrum 掃描的列數。
spectrum_scan_size_mb	bigint	Amazon S3 中 Amazon Redshift Spectrum 掃描的資料量 (MB)。
query_queue_time	bigint	查詢排入佇列的時間 (秒)。

SVL_QUERY_METRICS_SUMMARY

SVL_QUERY_METRICS_SUMMARY 檢視顯示已完成之查詢的指標最大值。此檢視衍生自 STL_QUERY_METRICS 系統資料表。請使用此檢視中的值來協助決定臨界值，以定義查詢監控規則。如需 Amazon Redshift 查詢監控之規則和指標的相關資訊，請參閱WLM 查詢監控規則。

所有使用者都可看見 SVL_QUERY_METRICS_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。

欄名稱	資料類型	描述
service_class	integer	WLM 查詢佇列 ID (服務類別)。查詢佇列定義於 WLM 組態。只會回報使用者定義之佇列的指標。如需服務類別 ID 的清單，請參閱 WLM 服務類別 ID。
query_cpu_time	bigint	查詢所用的 CPU 時間 (秒)。CPU 時間與查詢執行時間不同。
query_blocks_read	bigint	查詢所讀取的 1 MB 區塊數。
query_execution_time	bigint	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。
query_cpu_usage_percent	numeric(38,2)	查詢所用的 CPU 容量百分比。
query_temp_blocks_to_disk	bigint	查詢用來寫入中繼結果之磁碟空間量 (MB)。
segment_execution_time	bigint	單一區段經過的執行時間 (秒)。
cpu_skew	numeric(38,2)	任何分割 CPU 用量與所有分割之平均 CPU 用量的比率。此指標在區段層級中定義。
io_skew	numeric(38,2)	任何分割讀取區塊數上限 (I/O) 與所有分割平均讀取區塊數的比率。
scan_row_count	bigint	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。
join_row_count	bigint	聯結步驟中處理的資料列數。
nested_loop_join_row_count	bigint	巢狀迴圈聯結中的資料列數。

欄名稱	資料類型	描述
return_row_count	bigint	查詢傳回的資料列數。
spectrum_scan_row_count	bigint	Amazon S3 中的 Amazon Redshift Spectrum 掃描的列數。
spectrum_scan_size_mb	bigint	Amazon S3 中 Amazon Redshift Spectrum 掃描的資料量 (MB)。
query_queue_time	bigint	查詢排入佇列的時間 (秒)。

SVL_QUERY_QUEUE_INFO

摘要說明在工作負載管理 (WLM) 查詢佇列或遞交佇列中花費時間之查詢的詳細資訊。

SVL_QUERY_QUEUE_INFO 檢視會篩選系統執行的查詢並僅顯示使用者執行的查詢。

SVL_QUERY_QUEUE_INFO 檢視會摘要從 STL_QUERY、STL_WLM_QUERY 和 STL_COMMIT_STATS 系統資料表中的資訊。

只有超級使用者可以看到 SVL_QUERY_QUEUE_INFO。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
database	text	當查詢發出時，要將使用者連接至其中的資料庫名稱。
query	integer	查詢 ID。
xid	bigint	交易 ID。
userid	integer	產生查詢的使用者之 ID。
querytxt	text	查詢文字的前 100 個字元。

欄名稱	資料類型	描述
queue_start_time	timestamp	查詢進入 WLM 佇列的時間 (以 UTC 表示)。
exec_start_time	timestamp	查詢執行開始的時間 (以 UTC 表示)。
service_class	integer	服務類別的 ID。服務類別定義於 WLM 組態檔案。
slots	integer	WLM 查詢插槽的數量。
queue_elapsed	bigint	查詢在 WLM 佇列中等待的時間 (秒)。
exec_elapsed	bigint	花費在執行查詢的時間 (秒)。
wlm_total_elapsed	bigint	查詢在 WLM 佇列所花的時間 (queue_elapsed)，加上執行查詢所花的時間 (exec_elapsed)。
commit_queue_elapsed	bigint	查詢在遞交佇列中等待的時間 (秒)。
commit_exec_time	bigint	查詢在遞交操作中所花的時間 (秒)。
service_class_name	character(64)	服務類別的名稱。

範例查詢

下列範例為查詢在 WLM 佇列中所花的時間。

```
select query, service_class, queue_elapsed, exec_elapsed, wlm_total_elapsed
from svl_query_queue_info
where wlm_total_elapsed > 0;
```

```
  query  | service_class | queue_elapsed | exec_elapsed | wlm_total_elapsed
---------+---------------+---------------+--------------+-------------------
 2742669 |             6 |             2 |          916 |               918
 2742668 |             6 |             4 |          197 |               201
(2 rows)
```

SVL_QUERY_REPORT

Amazon Redshift 從各種 Amazon Redshift STL 系統資料表的 UNION 中建立 SVL_QUERY_REPORT 檢視，來提供已完成查詢步驟的資訊。

此檢視會依分割和依步驟來劃分已完成查詢的資訊，此有助於針對在 Amazon Redshift 叢集中的節點和分割問題進行疑難排解。

所有使用者都可看見 SVL_QUERY_REPORT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
分割	integer	執行步驟的資料分割。
segment	integer	區段號碼。 查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	已完成的查詢步驟。
start_time	timestamp	區段開始執行的確切時間 (以 UTC 表示)，精確度為 6 位數的小數秒。例如：**2012-12-12 11:29:19.131358**

欄名稱	資料類型	描述
end_time	timestamp	區段完成執行的確切時間 (以 UTC 表示)，精確度為 6 位數的小數秒。例如：**2012-12-12 11:29:19.131467**
elapsed_time	bigint	區段執行的所需時間 (微秒)。
rows	bigint	步驟產生的資料列數 (每分割)。此數字表示因步驟執行而造成之分割的資料列數，而不是步驟接收或處理的資料列數。也就是說，此為歷經該步驟且傳遞至下一個步驟之資料列數。
bytes	bigint	步驟產生的位元組數 (每分割)。
label	char(256)	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 `scan tbl=100448 name =user`)。三位數資料表 ID 通常是指暫時性資料表的掃描。當您看見 `tbl=0` 時，通常是指常數值的掃描。
is_diskbased	character (1)	查詢的這個步驟是否以磁碟型操作方式執行：true (**t**) 或 false (**f**)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。此值為在執行期間為使用而配置的 query_working_mem 臨界值，而非實際所用的記憶體量
is_rrscan	character (1)	若為 true (**t**)，表示已在步驟上使用範圍限制掃描。
is_delayed_scan	character (1)	若為 true (**t**)，表示已在該步驟上使用延遲的掃描。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之後，但在套用使用者定義的查詢篩選條件之前已發出的資料列總數。

範例查詢

下列查詢示範查詢 ID 為 279 的查詢之傳回列的資料扭曲。使用此查詢來決定資料庫資料是否在資料倉儲叢集中的分割中平均分散：

```
select query, segment, step, max(rows), min(rows),
case when sum(rows) > 0
then ((cast(max(rows) -min(rows) as float)*count(rows))/sum(rows))
else 0 end
from svl_query_report
where query = 279
group by query, segment, step
order by segment, step;
```

此查詢應會傳回與下列範例輸出類似的資料：

```
query | segment | step |   max    |   min    |         case
------+---------+------+----------+----------+----------------------
279 |       0 |    0 | 19721687 | 19721687 |                    0
279 |       0 |    1 | 19721687 | 19721687 |                    0
279 |       1 |    0 |   986085 |   986084 | 1.01411202804304e-06
279 |       1 |    1 |   986085 |   986084 | 1.01411202804304e-06
279 |       1 |    4 |   986085 |   986084 | 1.01411202804304e-06
279 |       2 |    0 |  1775517 |   788460 |     1.00098637606408
279 |       2 |    2 |  1775517 |   788460 |     1.00098637606408
279 |       3 |    0 |  1775517 |   788460 |     1.00098637606408
279 |       3 |    2 |  1775517 |   788460 |     1.00098637606408
279 |       3 |    3 |  1775517 |   788460 |     1.00098637606408
279 |       4 |    0 |  1775517 |   788460 |     1.00098637606408
279 |       4 |    1 |  1775517 |   788460 |     1.00098637606408
279 |       4 |    2 |        1 |        1 |                    0
279 |       5 |    0 |        1 |        1 |                    0
279 |       5 |    1 |        1 |        1 |                    0
279 |       6 |    0 |       20 |       20 |                    0
279 |       6 |    1 |        1 |        1 |                    0
279 |       7 |    0 |        1 |        1 |                    0
279 |       7 |    1 |        0 |        0 |                    0
(19 rows)
```

SVL_QUERY_SUMMARY

使用 SVL_QUERY_SUMMARY 檢視來尋找查詢執行的一般相關資訊。

SVL_QUERY_SUMMARY 檢視包含 SVL_QUERY_REPORT 檢視的資料子集。請留意，會從所有節點彙總 SVL_QUERY_SUMMARY 中的資訊。

> **Note**
>
> SVL_QUERY_SUMMARY 檢視僅包含 Amazon Redshift 執行之查詢的相關資訊，不包含其他公用程式和 DDL 命令的相關資訊。如需 Amazon Redshift 所完成全部陳述式的完整清單和相關資訊 (包含 DDL 和公用程式命令)，則可查詢 SVL_STATEMENTTEXT 檢視。

所有使用者都可看見 SVL_QUERY_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需 SVCS_QUERY_SUMMARY 的相關資訊，請參閱 SVCS_QUERY_SUMMARY。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
stm	integer	串流：查詢的並行區段集。一個查詢擁有一或多個串流。
seg	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	執行的查詢步驟。
maxtime	bigint	此步驟執行的時間量上限 (微秒)。
avgtime	bigint	此步驟執行的平均時間 (微秒)。
rows	bigint	查詢步驟中包含的資料列數。
位元組	bigint	查詢步驟中包含的資料位元組數。
rate_row	double precision	每資料列的查詢執行率。

欄名稱	資料類型	描述
rate_byte	double precision	每位元組的查詢執行率。
label	text	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 scan tbl=100448 name =user)。三位數資料表 ID 通常是指暫時性資料表的掃描。當您看見 `tbl=0` 時，通常是指常數值的掃描。
is_diskbased	character(1)	此查詢步驟在叢集的任何節點上是否以磁碟型操作方式執行：true (**t**) 或 false (**f**)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。
is_rrscan	character(1)	若為 true (**t**)，表示已在步驟上使用範圍限制掃描。預設為 false (**f**)。
is_delayed_scan	character(1)	若為 true (**t**)，表示已在該步驟上使用延遲的掃描。預設為 false (**f**)。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之前已發出的資料列總數。

範例查詢

檢視查詢步驟的處理資訊

下列查詢顯示查詢 87 每個步驟的基本處理資訊：

```
select query, stm, seg, step, rows, bytes
from svl_query_summary
where query = 87
order by query, seg, step;
```

此查詢會擷取查詢 87 的相關處理資訊，如下範例輸出所示：

```
 query | stm | seg | step |  rows  |  bytes
```

```
-------+-----+-----+------+--------+---------
87     |   0 |   0 |    0 |     90 |    1890
87     |   0 |   0 |    2 |     90 |     360
87     |   0 |   1 |    0 |     90 |     360
87     |   0 |   1 |    2 |     90 |    1440
87     |   1 |   2 |    0 | 210494 | 4209880
87     |   1 |   2 |    3 |  89500 |       0
87     |   1 |   2 |    6 |      4 |      96
87     |   2 |   3 |    0 |      4 |      96
87     |   2 |   3 |    1 |      4 |      96
87     |   2 |   4 |    0 |      4 |      96
87     |   2 |   4 |    1 |      1 |      24
87     |   3 |   5 |    0 |      1 |      24
87     |   3 |   5 |    4 |      0 |       0
(13 rows)
```

判斷查詢步驟是否溢出至磁碟

下列查詢顯示查詢 ID 為 1025 的任何查詢步驟 (請參閱 SVL_QLOG 檢視來了解如何取得查詢的查詢 ID) 是否溢出至磁碟或查詢完全在記憶體內執行：

```
select query, step, rows, workmem, label, is_diskbased
from svl_query_summary
where query = 1025
order by workmem desc;
```

此查詢傳回下列範例輸出：

```
query| step|  rows  |  workmem   |  label        | is_diskbased
-----+-----+--------+------------+---------------+--------------
1025 |  0  |16000000|  141557760 |scan tbl=9     | f
1025 |  2  |16000000|  135266304 |hash tbl=142   | t
1025 |  0  |16000000|  128974848 |scan tbl=116536| f
1025 |  2  |16000000|  122683392 |dist           | f
(4 rows)
```

透過掃描 IS_DISKBASED 的值，您可以檢視哪一個查詢步驟使用磁碟。針對查詢 1025，雜湊步驟在磁碟上執行。步驟可能在包含雜湊、彙總或排序步驟的磁碟上執行。若要只檢視磁碟型的查詢步驟，將 **`and is_diskbased = 't'`** 子句新增至以上範例的 SQL 陳述式。

SVL 還原 _ 更改表 _ 進度

在將傳統調整為 RA3 節點的大小時，使用 SVL_RESTORE_ALTER_TABLE_PROGRESS 來監視叢集中每個表格的移轉進度。它會在調整大小作業期間擷取資料遷移的歷史輸送量。有關經典調整為 RA3 節點的更多信息，請轉到經典調整大小。

SVL_ 還原 _ 更改表格 _ 進度僅對超級使用者可見。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_RESTORE_LOG 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

> **Note**
>
> 進度為100.00%或ABORTED在 7 天後刪除的資料列。在傳統重新調整大小期間或之後放置的表格資料列仍可顯示在 SVL_RESTOR_ALTER_TABLE_PROGRESS 中。

資料表欄

欄名稱	資料類型	描述
tbl	integer	資料表的 ID。
進度	char(32)	資料表重新分佈進度的狀態。可能的值為「0.00%到」100.00% 和「訊息」的百分比ABORTED。 ABORTED意味著重新分配在沒有完成的情況下停止，原因已在message列中解釋。
message	char(256)	與資料表重新發佈進度相關聯的訊息。

範例查詢

下列查詢會傳回執行中和佇列中的查詢。

```
select * from svl_restore_alter_table_progress;

tbl     | progress |                            message
--------+----------+-----------------------------------------------------------------
105614  | ABORTED  | Abort:Table no longer contains the prior dist key column.
105610  | ABORTED  | Abort:Table no longer contains the prior dist key column.
```

```
105594  | 0.00%    | Table waiting for alter diststyle conversion.
105602  | ABORTED  | Abort:Table no longer contains the prior dist key column.
105606  | ABORTED  | Abort:Table no longer contains the prior dist key column.
105598  | 100.00%  | Restored to distkey successfully.
```

SVL_S3LIST

使用 SVL_S3LIST 檢視以取得有關區段層級的 Amazon Redshift Spectrum 查詢詳細資訊。

所有使用者都可看見 SVL_S3LIST。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> SVL_S3LIST 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_EXTERNAL_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。
segment	integer	區段號碼。一個查詢包含多個區段。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
eventtime	timestamp	記錄事件的時間 (以 UTC 表示)。
儲存貯體	text	Amazon S3 儲存貯體名稱。
字首	text	Amazon S3 儲存貯體位置的字首。
recursive	char(1)	是否對子資料夾遞迴掃描。
retrieved_files	integer	所列檔案的數量。

欄名稱	資料類型	描述
max_file_size	bigint	所列檔案的檔案大小上限。
avg_file_size	double precision	所列檔案的平均檔案大小上限。
generated_splits	integer	檔案分割的數量。
avg_split_length	double precision	檔案分割的平均長度 (位元組)。
duration	bigint	檔案清單的期間 (微秒)。

範例查詢

以下範例會查詢前次執行查詢的 SVL_S3LIST。

```
select *
from svl_s3list
where query = pg_last_query_id()
order by query,segment;
```

SVL_S3LOG

使用 SVL_S3LOG 檢視以取得有關區段和節點分割層級的 Amazon Redshift Spectrum 查詢詳細資訊。

所有使用者都可看見 SVL_S3LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> SVL_S3LOG 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_EXTERNAL_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
pid	integer	處理程序 ID。
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
eventtime	timestamp	步驟開始執行的時間 (以 UTC 表示)。
message	text	日誌項目訊息。

範例查詢

以下範例會查詢前次執行查詢的 SVL_S3LOG。

```
select *
from svl_s3log
where query = pg_last_query_id()
order by query,segment,slice;
```

SVL_S3PARTITION

使用 SVL_S3PARTITION 檢視以取得有關區段和節點分割層級的 Amazon Redshift Spectrum 分割區詳細資訊。

所有使用者都可看見 SVL_S3PARTITION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> SVL_S3PARTITION 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_EXTERNAL_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
starttime	沒有時區的時間戳記	分割區清除開始執行的時間 (以 UTC 表示)。
endtime	沒有時區的時間戳記	分割區清除完成的時間 (以 UTC 表示)。
duration	bigint	經過時間 (微秒)。
total_partitions	integer	分割區總數。
qualified_partitions	integer	合格分割區數。
assigned_partitions	integer	分割上指定分割區數。
指派	character	指派的類型。

範例查詢

以下範例會取得前次完成查詢的分割區詳細資訊。

```
SELECT query, segment,
       MIN(starttime) AS starttime,
       MAX(endtime) AS endtime,
       datediff(ms,MIN(starttime),MAX(endtime)) AS dur_ms,
       MAX(total_partitions) AS total_partitions,
       MAX(qualified_partitions) AS qualified_partitions,
       MAX(assignment) as assignment_type
FROM svl_s3partition
WHERE query=pg_last_query_id()
GROUP BY query, segment
```

```
query | segment |           starttime           |           endtime           | dur_ms|
 total_partitions | qualified_partitions | assignment_type
------+---------+-------------------------------+-----------------------------+-------
+------------------+----------------------+---------------
99232 |       0 | 2018-04-17 22:43:50.201515    | 2018-04-17 22:43:54.674595  |  4473 |
       2526       |        334           | p
```

SVL_S3PARTITION_SUMMARY

使用 SVL_S3PARTITION_SUMMARY 檢視來取得區段層級的 Redshift Spectrum 查詢分割處理的摘要。

所有使用者都可看見 SVL_S3PARTITION_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

如需 SVCS_S3PARTITION 的相關資訊，請參閱 SVCS_S3PARTITION_SUMMARY。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
segment	integer	區段號碼。一個查詢包含多個區段。
指派	char(1)	節點間分割指派的類型。

欄名稱	資料類型	描述
min_starttime	timestamp	分割區處理開始的時間 (以 UTC 表示)。
max_endtime	timestamp	分割區處理完成的時間 (以 UTC 表示)。
min_duration	bigint	節點針對此查詢所使用的最小分割處理時間 (微秒)。
max_duration	bigint	節點針對此查詢所使用的最大分割處理時間 (微秒)。
avg_duration	bigint	節點針對此查詢所使用的平均分割處理時間 (微秒)。
total_partitions	integer	外部資料表中分割區的總數。
qualified_partitions	integer	合格分割區總數。
min_assigned_partitions	integer	在一個節點上指派的最小分割區數量。
max_assigned_partitions	integer	在一個節點上指派的最大分割區數量。
avg_assigned_partitions	bigint	在一個節點上指派的平均分割區數量。

範例查詢

以下範例會取得前次完成查詢的分割區掃描詳細資訊。

```
select query, segment, assignment, min_starttime, max_endtime, min_duration,
 avg_duration
from svl_s3partition_summary
where query = pg_last_query_id()
order by query,segment;
```

SVL_S3QUERY

使用 SVL_S3QUERY 檢視以取得有關區段和節點分割層級的 Amazon Redshift Spectrum 查詢詳細資訊。

所有使用者都可看見 SVL_S3QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

> **Note**
>
> SVL_S3QUERY 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 SYS_EXTERNAL_QUERY_DETAIL。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生給定項目的使用者 ID。
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
starttime	timestamp	查詢開始執行的時間 (以 UTC 表示)。

欄名稱	資料類型	描述
endtime	timestamp	查詢執行完成的時間 (以 UTC 表示)
elapsed	integer	經過時間 (微秒)。
external_table_name	char(136)	s3 掃描步驟之外部資料表名稱的內部格式。
is_partitioned	char(1)	若為 true (**t**)，此欄位值表示外部資料表已遭分割。
is_rrscan	char(1)	若為 true (**t**)，此欄位值表示已套用範圍限定的掃描。
s3_scanned_rows	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的列數。
s3_scanned_bytes	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的位元組數。
s3query_returned_rows	bigint	從 Redshift Spectrum 層傳回至叢集的資料列數。
s3query_returned_bytes	bigint	從 Redshift Spectrum 層傳回至叢集的位元組數。
files	integer	在此分割上此 S3 掃描步驟已處理的檔案數。
splits	int	在此分割上處理的分割數。有了大型可分割資料檔案 (例如，大於 512 MB 的資料檔案)，Redshift Spectrum 會嘗試將檔案分割至多個 S3 請求來進行平行處理。
total_split_size	bigint	在此分割上處理的所有分割大小總計 (位元組)。
max_split_size	bigint	為此分割處理的分割大小上限 (位元組)。

欄名稱	資料類型	描述
total_retries	integer	處理檔案的重試數總計。
max_retries	integer	個別處理檔案的重試數上限。
max_request_duration	integer	個別 Redshift Spectrum 請求的最大持續期間 (微秒)。
avg_request_duration	double precision	Redshift Spectrum 請求的平均持續時間 (微秒)。
max_request_parallelism	integer	在此 S3 掃描步驟的分割上未完成 Redshift Spectrum 數上限。
avg_request_parallelism	double precision	在此 S3 掃描步驟的分割上平行 Redshift Spectrum 請求數平均。

範例查詢

以下範例會取得前次完成查詢的掃描步驟詳細資訊。

```
select query, segment, slice, elapsed, s3_scanned_rows, s3_scanned_bytes,
 s3query_returned_rows, s3query_returned_bytes, files
from svl_s3query
where query = pg_last_query_id()
order by query,segment,slice;
```

```
query | segment | slice | elapsed | s3_scanned_rows | s3_scanned_bytes |
 s3query_returned_rows | s3query_returned_bytes | files
------+---------+-------+---------+-----------------+------------------
+----------------------+------------------------+------
 4587 |       2 |     0 |   67811 |               0 |                0 |
      0 |                      0 |     0
 4587 |       2 |     1 |  591568 |          172462 |         11260097 |
   8513 |                 170260 |     1
```

```
 4587 |       2 |     2 |  216849 |                0 |                0 |
      0 |                       0 |     0
 4587 |       2 |     3 |  216671 |                0 |                0 |
      0 |                       0 |     0
```

SVL_S3QUERY_SUMMARY

使用 SVL_S3QUERY_SUMMARY 檢視以取得在系統上執行之所有 Amazon Redshift Spectrum 查詢 (S3 查詢) 摘要。SVL_S3QUERY_SUMMARY 會在區段層級從 SVL_S3QUERY 彙總詳細資訊。

所有使用者都可看見 SVL_S3QUERY_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_EXTERNAL_QUERY_DETAIL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需 SVCS_S3QUERY_SUMMARY 的相關資訊，請參閱 SVCS_S3QUERY_SUMMARY。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行給定項目的使用者 ID。
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始執行的時間 (以 UTC 表示)。
endtime	timestamp	查詢完成的時間 (以 UTC 表示)。
elapsed	integer	查詢執行的所需時間長度 (微秒)。

欄名稱	資料類型	描述
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 **1**。如果查詢執行至完成，欄位會包含 **0**。
external_table_name	char(136)	資料表外部名稱之名稱的內部格式，適用於外部資料表掃描。
file_format	character(16)	外部資料表資料的檔案格式。
is_partitioned	char(1)	若為 true (**t**)，此欄位值表示外部資料表已遭分割。
is_rrscan	char(1)	若為 true (**t**)，此欄位值表示已套用範圍限定的掃描。
is_nested	char(1)	若為 true (**t**)，此欄位值表示已存取巢狀資料欄資料類型。
s3_scanned_rows	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的列數。
s3_scanned_bytes	bigint	根據壓縮檔案，從 Amazon S3 經掃描並送至 Redshift Spectrum 層的位元組數。
s3query_returned_rows	bigint	從 Redshift Spectrum 層傳回至叢集的資料列數。
s3query_returned_bytes	bigint	從 Redshift Spectrum 層傳回至叢集的位元組數。傳回至 Amazon Redshift 的大量資料可能會影響系統效能。
files	integer	此 Redshift Spectrum 查詢已處理的檔案數。限制平行處理之好處的少數檔案。
files_max	integer	在某分割上處理的檔案數上限。
files_avg	integer	在某分割上處理的平均檔案數。

欄名稱	資料類型	描述
splits	int	為此區段處理的分割數。在此分割上處理的分割數。有了大型可分割資料檔案 (例如，大於 512 MB 的資料檔案)，Redshift Spectrum 會嘗試將檔案分割至多個 S3 請求來進行平行處理。
splits_max	int	在此分割上處理的分割數上限。
splits_avg	int	在此分割上處理的平均分割數。
total_split_size	bigint	處理的所有分割大小總計。
max_split_size	bigint	處理的分割大小上限 (位元組)。
avg_split_size	bigint	處理的平均分割大小 (位元組)。
total_retries	integer	個別處理檔案的重試數總計。
max_retries	integer	任何處理檔案的重試數上限。
max_request_duration	integer	個別檔案請求的最大持續期間 (微秒)。長時間執行的查詢可能表示存在瓶頸。
avg_request_duration	double precision	檔案請求的平均持續時間 (微秒)。
max_request_parallelism	integer	此 Redshift Spectrum 查詢在某個分割上的最大平行請求數。
avg_request_parallelism	double precision	此 Redshift Spectrum 查詢在某個分割上的平均平行請求數。

欄名稱	資料類型	描述
total_slowdown_count	bigint	外部資料表掃描期間發生，具有降速錯誤的 Amazon S3 請求總數。
max_slowdown_count	integer	在某分割上外部資料表掃描期間發生，具有降速錯誤的最大 Amazon S3 請求數。

範例查詢

以下範例會取得前次完成查詢的掃描步驟詳細資訊。

```
select query, segment, elapsed, s3_scanned_rows, s3_scanned_bytes,
 s3query_returned_rows, s3query_returned_bytes, files
from svl_s3query_summary
where query = pg_last_query_id()
order by query,segment;
```

```
query | segment | elapsed | s3_scanned_rows | s3_scanned_bytes | s3query_returned_rows
 | s3query_returned_bytes | files
------+---------+---------+-----------------+------------------+----------------------
+------------------------+------
 4587 |       2 |   67811 |               0 |                0 |                      0
 |                      0 |     0
 4587 |       2 |  591568 |          172462 |         11260097 |                   8513
 |                 170260 |     1
 4587 |       2 |  216849 |               0 |                0 |                      0
 |                      0 |     0
 4587 |       2 |  216671 |               0 |                0 |                      0
 |                      0 |     0
```

SVL_S3RETRIES

使用 SVL_S3RETRIES 檢視來取得基於 Amazon S3 的 Amazon Redshift Spectrum 查詢失敗的資訊。

所有使用者都可看見 SVL_S3RETRIES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述		
query	integer	查詢 ID。		
segment	integer	區段號碼。 查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。		
節點	integer	節點號碼。		
分割	integer	特定區段執行依據的資料分割。		
eventtime	timestamp without time zone	步驟開始執行的時間 (以 UTC 表示)。		
retries	integer	查詢的重試次數。		
successful_fetches	integer	傳回資料的次數。		
file_size	bigint	檔案的大小 (以位元組為單位)。		
location	text	資料表的位置。		
message	text	錯誤訊息。		

範例查詢

以下範例會擷取關於失敗的 S3 查詢的資料。

```
SELECT svl_s3retries.query, svl_s3retries.segment, svl_s3retries.node,
 svl_s3retries.slice, svl_s3retries.eventtime, svl_s3retries.retries,
svl_s3retries.successful_fetches, svl_s3retries.file_size,
 btrim((svl_s3retries."location")::text) AS "location",
 btrim((svl_s3retries.message)::text)
AS message FROM svl_s3retries;
```

SVL_SPATIAL_SIMPLIFY

您可以使用 COPY 命令查詢系統檢視 SVL_SPATIAL_SIMPLIFY 以取得簡化空間幾何物件的資訊。當您在 Shapefile 上使用 COPY 時，您可以指定 SIMPLIFY `tolerance`、SIMPLIFY AUTO 和 SIMPLIFY AUTO `max_tolerance` 擷取選項。簡化的結果摘要在 SVL_SPATIAL_SIMPLIFY 系統檢視中。

設定 SIMPLIFY AUTO `max_tolerance` 時，此檢視會針對超出大小上限的每個幾何包含一列。設定 SIMPLIFY `tolerance` 時，會儲存整個 COPY 操作的一個列。此列參考 COPY 查詢 ID 和指定的簡化公差。

所有使用者都可看見 SVL_SPATIAL_SIMPLIFY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_SPATIAL_SIMPLIFY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
query	integer	產生此列的查詢 ID (COPY 命令)。
line_number	integer	當指定 COPY `SIMPLIFY AUTO` 選項時，這個值是 Shapefile 中簡化記錄的記錄編號。
maximum_tolerance	double	COPY 命令中指定的距離公差值。這是使用 `SIMPLIFY AUTO` 選項的最大公差值，或使用 `SIMPLIFY` 選項的固定公差值。
initial_size	integer	簡化之前 `GEOMETRY` 資料值的大小 (以位元組為單位)。
簡化	char(1)	指定 COPY `SIMPLIFY AUTO` 選項時，若已成功簡化幾何則為 `t`，否則為 `f`。如果使用給定的最大公差進行簡化後，

欄名稱	資料類型	描述
		幾何的大小仍大於幾何大小上限，則幾何可能無法成功簡化。
final_size	integer	指定 COPY SIMPLIFY AUTO 選項時，這是簡化後幾何的位元組大小。
final_tolerance	double	

範例查詢

下列查詢會傳回 COPY 簡化的記錄清單。

```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
 query | line_number | maximum_tolerance | initial_size | simplified | final_size |
 final_tolerance
-------+-------------+-------------------+--------------+------------+------------
+----------------------
    20 |     1184704 |                -1 |      1513736 | t          |    1008808 |
 1.276386653895e-05
    20 |     1664115 |                -1 |      1233456 | t          |    1023584 |
 6.11707814796635e-06
```

SVL_SPECTRUM_SCAN_ERROR

您可以查詢系統檢視 SVL_SPECTRUM_SCAN_ERROR，以取得有關 Redshift Spectrum 掃描錯誤的資訊。

所有使用者都可看見 SVL_SPECTRUM_SCAN_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_EXTERNAL_QUERY_ERROR 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

顯示記錄錯誤的範例。預設值是每個查詢 10 個項目。

欄名稱	資料類型	描述
userid	integer	產生此列之使用者的 ID。
query	integer	產生此列之查詢的 ID。
location	character(128)	要查詢之資料的位置。
rowid	character(128)	錯誤在檔案內的位置。rowid 部分以 : (冒號) 分隔，未來可能會添加其他部分。 `row_offset :row_group :row_id` Row_offset 是檔案中列的偏移量 (以位元組為單位)，針對不支援的檔案格式設定為 -1。資料表分為 row_groups，每個群組都有具有不同 row_id 的列。
colname	character(128)	查詢傳回的欄名稱。
original_value	character(128)	已查詢原始值。
modified_value	character(128)	根據查詢中指定的資料處理組態選項傳回的修改值。
觸發條件	character(128)	查詢中指定的資料處理選項。
動作	character(128)	與查詢中指定的資料處理選項相關聯的動作。
action_value	character(128)	與查詢中指定的資料處理選項相關聯的動作參數值。
error_code	integer	查詢中指定之資料處理選項的結果代碼。

範例查詢

下列查詢會傳回執行資料處理操作的列清單。

```
SELECT * FROM svl_spectrum_scan_error;
```

此查詢會傳回類似以下的結果。

```
  userid  query     location                                          rowid   colname
         original_value             modified_value       trigger          action
                 action_valueerror_code
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0     league_name
         Barclays Premier League    Barclays Premier Lea UNSPECIFIED      TRUNCATE
                                    156
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:0     league_nspi
          34595                     32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:1     league_nspi
          34151                     32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2     league_name
         Barclays Premier League    Barclays Premier Lea UNSPECIFIED      TRUNCATE
                                    156
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:2     league_nspi
          33223                     32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3     league_name
         Barclays Premier League    Barclays Premier Lea UNSPECIFIED      TRUNCATE
                                    156
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:3     league_nspi
          32808                     32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:4     league_nspi
          32790                     32767                UNSPECIFIED
OVERFLOW_VALUE                                    199
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:5     league_name
         Spanish Primera Division   Spanish Primera Divi UNSPECIFIED      TRUNCATE
                                    156
    100   1574007   s3://spectrum-uddh/league/spi_global_rankings.0:6     league_name
         Spanish Primera Division   Spanish Primera Divi UNSPECIFIED      TRUNCATE
                                    156
```

SVL_STATEMENTTEXT

使用 SVL_STATEMENTTEXT 檢視來取得在系統上執行之所有 SQL 命令的完整記錄。

SVL_STATEMENTTEXT 檢視包含 STL_DDLTEXT、STL_QUERYTEXT 和 STL_UTILITYTEXT 資料表中所有資料列的聯集。此檢視也包含對 STL_QUERY 資料表的聯結。

所有使用者都可看見 SVL_STATEMENTTEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	陳述式的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位為空白。
starttime	timestamp	陳述試開始執行的確切時間，精確度為 6 位數的小數秒。例如：**2009-06-12 11:29:19.131358**
endtime	timestamp	陳述試完成執行的確切時間，精確度為 6 位數的小數秒。例如：**2009-06-12 11:29:19.193640**
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
type	varchar(10)	SQL 陳述式的類型：**QUERY**、**DDL** 或 **UTILITY**。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回在 2009 年 6 月 16 日所執行的 DDL 陳述式：

```
select starttime, type, rtrim(text) from svl_statementtext
where starttime like '2009-06-16%' and type='DDL' order by starttime asc;

starttime                  | type |              rtrim
---------------------------|------|--------------------------------
2009-06-16 10:36:50.625097 | DDL  | create table ddltest(c1 int);
2009-06-16 15:02:16.006341 | DDL  | drop view alltickitjoin;
2009-06-16 15:02:23.65285  | DDL  | drop table sales;
2009-06-16 15:02:24.548928 | DDL  | drop table listing;
2009-06-16 15:02:25.536655 | DDL  | drop table event;
...
```

重建儲存的 SQL

若要重建儲存在 SVL_STATEMENTTEXT `text` 資料欄中的 SQL，則需執行 SELECT 陳述式以從 `text` 資料欄的一或多個部分建立 SQL。請先以新的一行取代任意 (\n) 特殊字元，再執行重建的 SQL。下列 SELECT 陳述式顯示的結果會是 `query_statement` 欄中重建的 SQL 資料列。

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS query_statement
from SVL_STATEMENTTEXT where pid=pg_backend_pid();
```

例如，下列查詢會選取 3 個資料欄。該查詢本身的長度超過 200 個字元，且會儲存在 SVL_STATEMENTTEXT 的多個部分中。

```
select
1 AS a01234567890123456789012345678901234567890123456789012345678901234567890,
2 AS b01234567890123456789012345678901234567890123456789012345678901234567890,
3 AS b01234567890123456789012345678901234567890123456789012345678901234
FROM stl_querytext;
```

在本範例中，查詢會儲存在 SVL_STATEMENTTEXT `text` 資料欄的 2 個部分 (資料列) 中。

```
select sequence, text from SVL_STATEMENTTEXT where pid = pg_backend_pid() order by
 starttime, sequence;
```

```
 sequence |
                 text

----------
+-----------------------------------------------------------------------------------------------
        0 | select\n1 AS
 a01234567890123456789012345678901234567890123456789012345678901234567890,\n2 AS
 b01234567890123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b01234567890123456789012345678901234567890123456789012345678901234
        1 | \nFROM stl_querytext;
```

若要重建儲存在 STL_STATEMENTTEXT 中的 SQL，請執行下列 SQL。

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
 within group (order by sequence) AS text
from SVL_STATEMENTTEXT where pid=pg_backend_pid();
```

若要在用戶端中使用產生的重建 SQL，請以新的一行取代任意 (\n) 特殊字元。

```


                    text

----------------------------------------------------------------------------------------------
 select\n1 AS a01234567890123456789012345678901234567890123456789012345678901234567890,
\n2 AS b01234567890123456789012345678901234567890123456789012345678901234567890,\n3 AS
 b01234567890123456789012345678901234567890123456789012345678901234\nFROM stl_querytext;
```

SVL_STORED_PROC_CALL

您可以查詢系統檢視 SVL_STORED_PROC_CALL，以取得預存程序呼叫的相關資訊，包括開始時間、結束時間及是否取消呼叫。每次預存程序呼叫會接收查詢 ID。

所有使用者都可看見 SVL_STORED_PROC_CALL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_PROCEDURE_CALL 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	使用者的 ID，該使用者的權限用於執行陳述式。如果此呼叫套疊於 SECURITY DEFINER 預存程序內，則此為該預存程序的擁有者的 userid。
session_userid	integer	使用者的 ID，該使用者建立工作階段，並且是最上層預存程序呼叫的叫用者。
query	integer	程序呼叫的查詢 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位值為預設值。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。通常，工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。在特定內部事件後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 pid 值。如需詳細資訊，請參閱 STL_RESTARTED_SESSIONS。
database	character(32)	當查詢發出時，使用者連接的資料庫名稱。
querytxt	character(4000)	程序呼叫查詢的實際文字。
starttime	timestamp	查詢開始執行的 UTC 時間，小數秒的精確度為 6 位數，例如：**2009-06-12 11:29:19.131358.**
endtime	timestamp	查詢完成執行的 UTC 時間，小數秒的精確度為 6 位數，例如：**2009-06-12 11:29:19.131358.**
aborted	integer	如果預存程序已被系統停止或被使用者取消，則此欄包含 1。如果呼叫執行至完成，則此欄包含 0。

欄名稱	資料類型	描述
from_sp_call	integer	如果程序呼叫由另一個程序呼叫所叫用，則此欄包含外層呼叫的查詢 ID。否則此欄位為 NULL。

範例查詢

下列查詢傳回過去一天的預存程序呼叫的經歷時間 (依遞減順序) 和完成狀態。

```
select query, datediff(seconds, starttime, endtime) as elapsed_time, aborted,
 trim(querytxt) as call from svl_stored_proc_call where starttime >= getdate() -
 interval '1 day' order by 2 desc;

  query | elapsed_time | aborted |                                                call
--------+--------------+---------
+-----------------------------------------------------------------------------------------
   4166 |            7 |       0 | call search_batch_status(35,'succeeded');
   2433 |            3 |       0 | call test_batch (123456)
   1810 |            1 |       0 | call prod_benchmark (123456)
   1836 |            1 |       0 | call prod_testing (123456)
   1808 |            1 |       0 | call prod_portfolio ('N', 123456)
   1816 |            1 |       1 | call prod_portfolio ('Y', 123456)
```

SVL_STORED_PROC_MESSAGES

您可以查詢系統檢視 SVL_STORED_PROC_MESSAGES 來取得預存程序訊息的相關資訊。即使預存程序呼叫遭到取消，引發的訊息還是會記錄到日誌。每次預存程序呼叫會接收查詢 ID。如需如何設定記錄訊息最小層級的相關資訊，請參閱 stored_proc_log_min_messages。

所有使用者都可看見 SVL_STORED_PROC_MESSAGES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_PROCEDURE_MESSAGES 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	使用者的 ID，該使用者的權限用於執行陳述式。如果此呼叫套疊於 SECURITY DEFINER 預存程序內，則此為該預存程序的擁有者的 userid。
session_userid	integer	使用者的 ID，該使用者建立工作階段，並且是最上層預存程序呼叫的叫用者。
pid	integer	程序 ID。
xid	bigint	程序呼叫查詢的交易 ID。
query	integer	程序呼叫的查詢 ID。
recordtime	timestamp	引發訊息的 UTC 時間。
loglevel	integer	引發訊息的日誌層級數值。可能的值：20 – 適用於 LOG 30 – 適用於 INFO 40 – 適用於 NOTICE 50 – 適用於 WARNING 60 – 適用於 EXCEPTION
loglevel_text	character(10)	對應到 loglevel 中數值的日誌層級。可能的值：LOG (記錄)、INFO (資訊)、NOTICE (通知)、WARNING (警告) 和 EXCEPTION (例外)。
message	character(1024)	引發訊息的文字。
linenum	integer	引發陳述式的行號。
querytext	character(500)	程序呼叫查詢的實際文字。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位值為預設值。
aborted	integer	如果預存程序已被系統停止或被使用者取消，則此欄包含 1。如果呼叫執行至完成，則此欄包含 0。

欄名稱	資料類型	描述
message_xid	bigint	提出訊息的交易 ID。

範例查詢

下列 SQL 陳述式示範如何使用 SVL_STORED_PROC_MESSAGES 來檢閱引發的訊息。

```
-- Create and run a stored procedure
CREATE OR REPLACE PROCEDURE test_proc1(f1 int) AS
$$
BEGIN
    RAISE INFO 'Log Level: Input f1 is %',f1;
    RAISE NOTICE 'Notice Level: Input f1 is %',f1;
    EXECUTE 'select invalid';
    RAISE NOTICE 'Should not print this';

EXCEPTION WHEN OTHERS THEN
     raise exception 'EXCEPTION level: Exception Handling';
END;
$$ LANGUAGE plpgsql;

-- Call this stored procedure
CALL test_proc1(2);

-- Show raised messages with level higher than INFO
SELECT query, recordtime, loglevel, loglevel_text, trim(message) as message, aborted
 FROM svl_stored_proc_messages
  WHERE loglevel > 30 AND query = 193 ORDER BY recordtime;

 query |         recordtime         | loglevel | loglevel_text |               message
              | aborted
-------+----------------------------+----------+---------------
+-------------------------------------+---------
   193 | 2020-03-17 23:57:18.277196 |       40 | NOTICE        | Notice Level: Input f1
 is 2         |       1
   193 | 2020-03-17 23:57:18.277987 |       60 | EXCEPTION     | EXCEPTION level:
 Exception Handling |       1
(2 rows)

-- Show raised messages at EXCEPTION level
```

```
SELECT query, recordtime, loglevel, loglevel_text, trim(message) as message, aborted
 FROM svl_stored_proc_messages
  WHERE loglevel_text = 'EXCEPTION' AND query = 193 ORDER BY recordtime;

 query |         recordtime         | loglevel | loglevel_text |               message
              | aborted
-------+----------------------------+----------+---------------
+-----------------------------------+---------
   193 | 2020-03-17 23:57:18.277987 |       60 | EXCEPTION     | EXCEPTION level:
 Exception Handling |       1
```

下列 SQL 陳述式示範如何使用 SVL_STORED_PROC_MESSAGES，在建立預存程序時使用 SET 選項來檢閱引發的訊息。因為 test_proc() 的最低日誌層級為 NOTICE，只有 NOTICE、WARNING 和 EXCEPTION 層級訊息會記錄在 SVL_STORED_PROC_MESSAGES 中。

```
-- Create a stored procedure with minimum log level of NOTICE
CREATE OR REPLACE PROCEDURE test_proc() AS
$$
BEGIN
    RAISE LOG 'Raise LOG messages';
    RAISE INFO 'Raise INFO messages';
    RAISE NOTICE 'Raise NOTICE messages';
    RAISE WARNING 'Raise WARNING messages';
    RAISE EXCEPTION 'Raise EXCEPTION messages';
    RAISE WARNING 'Raise WARNING messages again'; -- not reachable
END;
$$ LANGUAGE plpgsql SET stored_proc_log_min_messages = NOTICE;

-- Call this stored procedure
CALL test_proc();

-- Show the raised messages
SELECT query, recordtime, loglevel_text, trim(message) as message, aborted FROM
 svl_stored_proc_messages
  WHERE query = 149 ORDER BY recordtime;

 query |         recordtime         | loglevel_text |          message          |
 aborted
-------+----------------------------+---------------+---------------------------
+---------
   149 | 2020-03-16 21:51:54.847627 | NOTICE        | Raise NOTICE messages     |
 1
```

```
   149 | 2020-03-16 21:51:54.84766  | WARNING       | Raise WARNING messages   |
 1
   149 | 2020-03-16 21:51:54.847668 | EXCEPTION     | Raise EXCEPTION messages |
 1
(3 rows)
```

SVL_TERMINATE

記錄使用者取消或終止程序的時間。

SELECT PG_TERMINATE_BACKEND(pid)、SELECT PG_CANCEL_BACKEND(pid) 和 CANCEL pid 可在 SVL_TERMINATE 中建立日誌項目。

只有超級使用者可以看到 SVL_TERMINATE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_QUERY_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
pid	integer	已取消或終止之程序的程序 ID。
eventtime	timestamp	取消或終止程序的時間。
userid	integer	執行命令之使用者的使用者 ID。
type	string	終止的類型。它可以是 CANCEL 或 TERMINATE。

下列命令顯示最新的取消查詢。

```
select * from svl_terminate order by eventtime desc limit 1;
 pid  |         eventtime          | userid |  type
------+----------------------------+--------+--------
 8324 | 2020-03-24 09:42:07.298937 |      1 | CANCEL
(1 row)
```

SVL_UDF_LOG

記錄使用者定義函數 (UDF) 執行期間產生的系統定義的錯誤和警告訊息。

所有使用者都可看見 SVL_UDF_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_UDF_LOG 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
query	bigint	查詢 ID。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
message	char(4096)	函數產生的訊息。
已建立	timestamp	日誌建立的時間。
回溯	char(4096)	若果可用，此值會提供 UDF 的堆疊回溯。如需詳細資訊，請參閱 Python Standard Library 的 traceback。
funcname	character(256)	正在執行之 UDF 的名稱。
節點	integer	訊息產生的節點。
分割	integer	訊息產生的分割。
seq	integer	分割上訊息的順序。

範例查詢

以下範例說明 UDF 如何處理系統定義的錯誤。第一個區塊顯示傳回引數反向之 UDF 函數的定義。執行函數並提供 0 引數時，如第二個區塊所示，函數會傳回錯誤。第三個陳述式會讀取在 SVL_UDF_LOG 中記錄的錯誤訊息

```
-- Create a function to find the inverse of a number

CREATE OR REPLACE FUNCTION  f_udf_inv(a int)
  RETURNS float IMMUTABLE
AS $$
   return 1/a
$$ LANGUAGE plpythonu;

-- Run the function with a 0 argument to create an error
Select f_udf_inv(0) from sales;

-- Query SVL_UDF_LOG to view the message

Select query, created, message::varchar
from svl_udf_log;

 query |          created           | message
-------+----------------------------
+------------------------------------------------------------
  2211 | 2015-08-22 00:11:12.04819  | ZeroDivisionError: long division or modulo by
 zero\nNone
```

下列範例會將記錄且警告訊息新增至 UDF，以至於除以零操作會導致警告訊息，而不是停止並出現錯誤訊息。

```
-- Create a function to find the inverse of a number and log a warning

CREATE OR REPLACE FUNCTION f_udf_inv_log(a int)
  RETURNS float IMMUTABLE
 AS $$
  import logging
  logger = logging.getLogger() #get root logger
  if a==0:
    logger.warning('You attempted to divide by zero.\nReturning zero instead of error.
\n')
    return 0
  else:
     return 1/a
$$ LANGUAGE plpythonu;
```

下列範例會執行函數，接著查詢 SVL_UDF_LOG，來檢視訊息。

```
-- Run the function with a 0 argument to trigger the warning
Select f_udf_inv_log(0) from sales;

-- Query SVL_UDF_LOG to view the message

Select query, created, message::varchar
from svl_udf_log;

query |          created           | message
------+----------------------------+---------------------------------
    0 | 2015-08-22 00:11:12.04819  | You attempted to divide by zero.
                                     Returning zero instead of error.
```

SVL_USER_INFO

您可以使用 SVL_USER_INFO 檢視來擷取關於 Amazon Redshift 資料庫使用者的資料。

只有超級使用者可以看到 SVL_USER_INFO。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄

欄名稱	資料類型	描述
usename	text	角色的使用者名稱。
usesysid	integer	使用者的使用者 ID。
usecreatedb	boolean	一個值，指出使用者是否具有建立資料庫的許可。
usesuper	boolean	一個值，指出使用者是否為超級使用者。
usecatupd	boolean	一個值，指出使用者可以更新系統目錄。
useconnlimit	text	使用者可以開啟的連線數量。
syslogaccess	text	一個值，指出使用者是否具有系統日誌的存取權。兩個可能值為 RESTRICTED 和 UNRESTRICTED 。RESTRICTED 表示不是超級使用

欄名稱	資料類型	描述
		者的使用者可以看到其記錄。UNRESTRICTED 表示不是超級使用者的使用者可以看到他們具有 SELECT 權限的系統檢視和資料表中的所有記錄。
last_ddl_ts	timestamp	使用者執行的最新資料定義語言 (DDL) create 陳述式的時間戳記。
sessiontimeout	integer	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。如需叢集的閒置或非作用中逾時設定詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制。
external_id	text	第三方身分提供者中使用者的唯一識別碼。

範例查詢

下列命令會從 SVL_USER_INFO 擷取使用者資訊。

```
SELECT * FROM SVL_USER_INFO;
```

SVL_VACUUM_PERCENTAGE

SVL_VACUUM_PERCENTAGE 檢視報告在執行清空後為資料表配置的資料區塊百分比。此百分比數顯示磁碟空間的回收量。如需清空公用程式的相關資訊，請參閱 VACUUM 命令。

只有超級使用者可以看到 SVL_VACUUM_PERCENTAGE。如需詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

此資料表中的部份或所有資料也會在 SYS 監控檢視 SYS_VACUUM_HISTORY 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
xid	bigint	清空陳述式的交易 ID。
table_id	integer	清空資料表的資料表 ID。

欄名稱	資料類型	描述
百分比	bigint	清空後資料區塊的百分比 (與在清空執行前資料表中區塊數相對)。

範例查詢

下列查詢顯示在資料表 100238 上特定操作的百分比：

```
select * from svl_vacuum_percentage
where table_id=100238 and xid=2200;

xid  | table_id | percentage
-----+----------+------------
1337 |   100238 |         60
(1 row)
```

在此清空操作後，資料表包含原始區塊的 60%。

系統目錄資料表

主題

- PG_ATTRIBUTE_INFO
- PG_CLASS_INFO
- PG_DATABASE_INFO
- PG_DEFAULT_ACL
- PG_EXTERNAL_SCHEMA
- PG_LIBRARY
- PG_PROC_INFO
- PG_STATISTIC_INDICATOR
- PG_TABLE_DEF
- PG_USER_INFO
- 查詢目錄資料表

系統目錄存放結構描述中繼資料，例如資料表和資料欄的相關資訊。系統目錄資料表有 PG 字首。

Amazon Redshift 使用者可存取標準 PostgreSQL 目錄資料表。如需 PostgreSQL 系統目錄的詳細資訊，請參閱 PostgreSQL system tables

PG_ATTRIBUTE_INFO

PG_ATTRIBUTE_INFO 是以 PostgreSQL 目錄資料表 PG_ATTRIBUTE 和內部目錄資料表 PG_ATTRIBUTE_ACL 為基礎建立的 Amazon Redshift 系統檢視。PG_ATTRIBUTE_INFO 包含資料表或檢視的資料欄詳細資訊，其中包括資料欄存取控制清單 (如果有的話)。

資料表欄位

除了 PG_ATTRIBUTE 中的資料欄，PG_ATTRIBUTE_INFO 還會顯示下列幾個資料欄。

資料欄名稱	資料類型	描述
attacl	aclitem[]	在此資料欄上特別授與的資料欄層級存取權限 (如果有的話)。

PG_CLASS_INFO

PG_CLASS_INFO 是一個 Amazon Redshift 系統檢視，以 PostgreSQL 類別資料表 PG_CLASS 和 PG_CLASS_EXTENDED 為建置基礎。PG_CLASS_INFO 包含關於資料表建立時間和目前分佈樣式的詳細資訊。如需更多詳細資訊，請參閱 使用資料分佈樣式。

所有使用者都可看見 PG_CLASS_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需更多詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄位

除了 PG_CLASS 中的資料欄，PG_CLASS_INFO 會顯示下列資料欄。PG_CLASS 中的 `oid` 欄在 PG_CLAS_INFO 資料表中稱為 `relid`。

資料欄名稱	資料類型	描述
relcreationtime	timestamp	建立資料表的時間，以 UTC 表示。
releffectivediststyle	integer	資料表的分佈樣式，或如果資料表使用自動分佈，則為 Amazon Redshift 指派的目前分佈樣式。

PG_CLASS_INFO 中的 RELEFFECTIVEDISTSTYLE 欄指出資料表的目前分佈樣式。如果資料表使用自動分佈，則 RELEFFECTIVEDISTSTYLE 為 10、11 或 12，這指出有效分佈樣式為 AUTO (ALL)、AUTO (EVEN) 或 AUTO (KEY)。如果資料表使用自動分布，則分佈樣式一開始可能顯示 AUTO (ALL)，然後當資料表成長時變更為 AUTO (EVEN)，如果發現某欄可用作分佈索引鍵，則變更為 AUTO (KEY)。

下表提供 RELEFFECTIVEDISTSTYLE 欄中每個值的分佈樣式：

RELEFFECTIVEDISTSTYLE	目前分佈樣式
0	EVEN
1	KEY
8	ALL
10	AUTO (ALL)
11	AUTO (EVEN)
12	AUTO (KEY)

範例

下列查詢會傳回目錄中資料表的目前分佈樣式。

```
select reloid as tableid,trim(nspname) as schemaname,trim(relname) as
 tablename,reldiststyle,releffectivediststyle,
CASE WHEN "reldiststyle" = 0 THEN 'EVEN'::text
     WHEN "reldiststyle" = 1 THEN 'KEY'::text
     WHEN "reldiststyle" = 8 THEN 'ALL'::text
     WHEN "releffectivediststyle" = 10 THEN 'AUTO(ALL)'::text
     WHEN "releffectivediststyle" = 11 THEN 'AUTO(EVEN)'::text
     WHEN "releffectivediststyle" = 12 THEN 'AUTO(KEY)'::text ELSE '<<UNKNOWN>>'::text
 END as diststyle,relcreationtime
from pg_class_info a left join pg_namespace b on a.relnamespace=b.oid;
```

```
 tableid | schemaname | tablename | reldiststyle | releffectivediststyle | diststyle  |
      relcreationtime
```

```
---------+------------+-----------+-------------+----------------------+------------
+---------------------------
 3638033 | public     | customer  |           0 |                    0 | EVEN       |
 2019-06-13 15:02:50.666718
 3638037 | public     | sales     |           1 |                    1 | KEY        |
 2019-06-13 15:03:29.595007
 3638035 | public     | lineitem  |           8 |                    8 | ALL        |
 2019-06-13 15:03:01.378538
 3638039 | public     | product   |           9 |                   10 | AUTO(ALL)  |
 2019-06-13 15:03:42.691611
 3638041 | public     | shipping  |           9 |                   11 | AUTO(EVEN) |
 2019-06-13 15:03:53.69192
 3638043 | public     | support   |           9 |                   12 | AUTO(KEY)  |
 2019-06-13 15:03:59.120695
(6 rows)
```

PG_DATABASE_INFO

PG_DATABASE_INFO 是以 PostgreSQL 目錄資料表 PG_DATABASE 為延伸基礎的 Amazon Redshift 系統檢視。

所有使用者都可看見 PG_DATABASE_INFO。

資料表欄位

除了 PG_DATABASE 中的資料欄外，PG_DATABASE_INFO 還包含下列資料欄。PG_DATABASE 中的 `oid` 欄在 PG_DATABASE_INFO 資料表中稱為 `datid`。如需詳細資訊，請參閱 PostgreSQL 文件。

資料欄名稱	資料類型	描述
datid	oid	系統資料表內部使用的物件識別符 (OID)。
datconnlimit	text	您可以對此資料庫進行的並行連線數量上限。值為 -1 表示沒有限制。

PG_DEFAULT_ACL

存放預設存取權限的相關資訊。如需預設存取權限的詳細資訊，請參閱 ALTER DEFAULT PRIVILEGES。

所有使用者都可看見 PG_DEFAULT_ACL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需更多詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄位

資料欄名稱	資料類型	描述
defacluser	integer	將所列權限套用至其中的使用者 ID。
defaclnamespace	oid	將預設權限所套用至其中的結構描述物件 ID。如果未指定任何結構描述，預設值會是 0。
defaclobjtype	character	將預設權限所套用至其中的物件類型。有效值如下： • r–關聯 (資料表或檢視) • f–函數 • p–預存程序
defaclacl	aclitem[]	定義指定使用者或使用者群組的預設權限和物件類型的字串。 如果將此權限授予另一個使用者，字串格式如下： *{ username=privilegestring/grantor }* username 將權限授予至其中的使用者名稱。如果省略使用者名稱，則會將權限授予給 PUBLIC。 如果將此權限授予另一個使用者群組，字串格式如下： *{ "group groupname=privilegestring/grantor" }* privilegestring 指定授予哪一個權限的字串。 有效值為：

資料欄名稱	資料類型	描述
		• r–SELECT (讀取) • a–INSERT (附加) • w–UPDATE (寫入) • d–DELETE • x–授予建立外部索引鍵限制的權限 (REFERENCES)。 • X–EXECUTE • *–指出接收前述權限的使用者可以輪流將相同的權限授予給他人 (WITH GRANT OPTION)。 grantor 授予權限的使用者名稱。 下列範例指出，使用者 admin 將所有權限 (包含 WITH GRANT OPTION) 授予給使用者 dbuser。 `dbuser=r*a*w*d*x*X*/admin`

範例

下列查詢會傳回為資料庫定義之所有預設權限。

```
select pg_get_userbyid(d.defacluser) as user,
n.nspname as schema,
case d.defaclobjtype when 'r' then 'tables' when 'f' then 'functions' end
as object_type,
array_to_string(d.defaclacl, ' + ')  as default_privileges
from pg_catalog.pg_default_acl d
left join pg_catalog.pg_namespace n on n.oid = d.defaclnamespace;

 user  | schema | object_type |              default_privileges
-------+--------+-------------+-------------------------------------------------------
 admin | tickit | tables      | user1=r/admin + "group group1=a/admin" + user2=w/admin
```

前述範例中的結果顯示針對在 admin 結構描述中的使用者 tickit 所建立的所有新資料表，admin 將 SELECT 權限授予給 user1、INSERT 權限授予給 group1 而 UPDATE 權限授予給 user2。

PG_EXTERNAL_SCHEMA

存放外部結構描述的相關資訊。

所有使用者都可看見 PG_EXTERNAL_SCHEMA。超級使用者可以看見所有資料列；一般使用者只能看見它們有權存取的中繼資料。如需更多詳細資訊，請參閱 CREATE EXTERNAL SCHEMA。

資料表欄位

資料欄名稱	資料類型	描述
esoid	oid	外部結構描述 ID。
eskind	integer	一種外部結構描述。
esdbname	text	外部資料庫名稱。
esoptions	text	外部結構描述選項。

範例

下列範例顯示外部結構描述的詳細資訊。

```
select esoid, nspname as schemaname, nspowner, esdbname as external_db, esoptions
from pg_namespace a,pg_external_schema b where a.oid=b.esoid;

esoid  | schemaname      | nspowner | external_db | esoptions

-------+-----------------+----------+-------------
+-------------------------------------------------------------
100134 | spectrum_schema |      100 | spectrum_db |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
100135 | spectrum        |      100 | spectrumdb  |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
100149 | external        |      100 | external_db |
 {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

PG_LIBRARY

存放使用者定義之資料庫的相關資訊。

所有使用者都可看見 PG_LIBRARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需更多詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄位

資料欄名稱	資料類型	描述
name	name	資料庫名稱。
language_oid	oid	保留以供系統使用。
file_store_id	integer	保留以供系統使用。
owner	integer	資料庫擁有者的使用者 ID。

範例

以下範例會傳回使用者安裝資料庫的資訊。

```
select * from pg_library;

name       | language_oid | file_store_id | owner
-----------+--------------+---------------+------
f_urlparse |       108254 |          2000 |   100
```

PG_PROC_INFO

PG_PROC_INFO 是以 PostgreSQL 目錄資料表 PG_PROC 和內部目錄資料表 PG_PROC_EXTENDED 為基礎建立的 Amazon Redshift 系統檢視。PG_PROC_INFO 包含儲存程序和函數的詳細資訊，包括輸出引數 (若有) 的相關資訊。

資料表欄位

除了 PG_PROG 中的欄，PG_PROC_INFO 還會顯示下列幾欄。PG_PROC 中的 oid 欄在 PG_PROC_INFO 資料表中稱為 prooid。

資料欄名稱	資料類型	描述
prooid	oid	函數或預存程序的物件 ID。
prokind	"char"	指出函數或預存程序類型的值。此值為 'f' 代表一般函數、'p' 代表預存程序、'a' 代表彙總函數。
proargmodes	"char"[ ]	由程序引數的模式組成的陣列，編碼為 'i' 代表 IN 引數、'o' 代表 OUT 引數、'b' 代表 INOUT 引數。如果所有引數都是 IN 引數，此欄位為 NULL。與 proallargtypes 陣列中的位置對應的下標。
proallargtypes	oid[ ]	由程序引數的資料類型組成的陣列。此陣列包含所有類型的引數 (包括 OUT 和 INOUT 引數)。不過，如果所有引數都是 IN 引數，此欄位為 NULL。下標以 1 開始。相反地，proargtypes 中的下標以 0 開始。

PG_PROC_INFO 中的欄位 proargnames 包含所有類型的引數 (包括 OUT 和 INOUT) 的名稱 (若有)。

PG_STATISTIC_INDICATOR

存放自前次 ANALYZE 以來插入或刪除的列數之相關資訊。PG_STATISTIC_INDICATOR 資料表會在下列 DML 操作後頻繁的更新，因此統計資訊是近似值。

只有超級使用者可以看到 PG_STATISTIC_INDICATOR。如需更多詳細資訊，請參閱 系統資料表和檢視中資料的可見性。

資料表欄位

資料欄名稱	資料類型	描述
stairelid	oid	資料表 ID
stairows	float	資料表中列總數。

資料欄名稱	資料類型	描述
staiins	float	自前次 ANALYZE 以來插入的列數。
staidels	float	自前次 ANALYZE 以來刪除或更新的列數。

範例

以下範例會傳回自前次 ANALYZE 以來資料表變更的資訊。

```
select * from pg_statistic_indicator;

stairelid | stairows | staiins | staidels
----------+----------+---------+---------
   108271 |       11 |       0 |        0
   108275 |      365 |       0 |        0
   108278 |     8798 |       0 |        0
   108280 |    91865 |       0 |   100632
   108267 |    89981 |   49990 |     9999
   108269 |      808 |     606 |      374
   108282 |   152220 |   76110 |   248566
```

PG_TABLE_DEF

存放資料表欄位的相關資訊。

PG_TABLE_DEF 僅傳回使用者可見之資料表的相關資訊。如果 PG_TABLE_DEF 未傳回預期的結果，確認 search_path 參數是否正確設定為包含相關的結構描述。

您可以使用 SVV_TABLE_INFO 來檢視資料表的更全面性資訊，包括資料配送偏度、索引鍵配送偏度、資料表大小、統計等方面的問題。

資料表欄位

資料欄名稱	資料類型	描述
結構描述名稱	name	結構描述名稱。

資料欄名稱	資料類型	描述
資料表名稱	name	資料表名稱
欄位	name	資料欄名稱。
類型	text	欄位資料類型。
編碼	character(32)	欄位編碼。
distkey	布林值	若此欄位是資料表的分佈索引鍵，則為 true。
sortkey	integer	排序索引鍵中的欄位順序。如果資料表使用的是複合排序索引鍵，則排序索引鍵部分的所有欄位會有正值，指示排序索引鍵中欄位的位置。如果資料表使用的是交錯的排序索引鍵，則排序索引鍵部分的每個欄會有交替為正或負的值，其中絕對值表示排序索引鍵中的欄位置。若為 0，則欄位不是排序索引鍵的一部分。
notnull	布林值	如果欄位有 NOT NULL 限制，則為 true。

範例

下列範例顯示 LINEORDER_COMPOUND 資料表的複合排序索引鍵欄位。

```
select "column", type, encoding, distkey, sortkey, "notnull"
from pg_table_def
where tablename = 'lineorder_compound'
and sortkey <> 0;

column       | type    | encoding | distkey | sortkey | notnull
-------------+---------+----------+---------+---------+--------
lo_orderkey  | integer | delta32k | false   |       1 | true
lo_custkey   | integer | none     | false   |       2 | true
lo_partkey   | integer | none     | true    |       3 | true
lo_suppkey   | integer | delta32k | false   |       4 | true
lo_orderdate | integer | delta    | false   |       5 | true
(5 rows)
```

下列範例顯示 LINEORDER_INTERLEAVED 資料表的交錯排序索引鍵欄位。

```
select "column", type, encoding, distkey, sortkey, "notnull"
from pg_table_def
where tablename = 'lineorder_interleaved'
and sortkey <> 0;

column       | type    | encoding | distkey | sortkey | notnull
-------------+---------+----------+---------+---------+--------
lo_orderkey  | integer | delta32k | false   |      -1 | true
lo_custkey   | integer | none     | false   |       2 | true
lo_partkey   | integer | none     | true    |      -3 | true
lo_suppkey   | integer | delta32k | false   |       4 | true
lo_orderdate | integer | delta    | false   |      -5 | true
(5 rows)
```

PG_TABLE_DEF 將僅傳回在搜尋路徑中包含之結構描述中資料表的相關資訊。如需更多詳細資訊，請參閱 search_path。

例如，假設您建立新結構描述和新資料表，以及查詢 PG_TABLE_DEF。

```
create schema demo;
create table demo.demotable (one int);
select * from pg_table_def where tablename = 'demotable';

schemaname|tablename|column| type | encoding | distkey | sortkey | notnull
----------+---------+------+------+----------+---------+---------+--------
```

查詢未傳回新資料表的任何列。檢查 search_path 的設定。

```
show search_path;

  search_path
---------------
 $user, public
(1 row)
```

將 demo 結構描述新增至搜尋路徑中並再次執行查詢。

```
set search_path to '$user', 'public', 'demo';

select * from pg_table_def where tablename = 'demotable';

schemaname| tablename |column|  type   | encoding |distkey|sortkey| notnull
```

```
----------+-----------+------+---------+----------+-------+-------+--------
demo      | demotable | one  | integer | none     | f     |     0 | f
(1 row)
```

PG_USER_INFO

PG_USER_INFO 是 Amazon Redshift 系統檢視，用於顯示使用者訊息，例如使用者 ID 和密碼過期時間。

只有超級使用者可以看到 PG_USER_INFO。

資料表欄位

PG_USER_INFO 包含下列欄。如需詳細資訊，請參閱 PostgreSQL 文件。

資料欄名稱	資料類型	描述
usename	name	使用者名稱。
usesysid	integer	使用者 ID。
usecreatedb	boolean	如果使用者可以建立資料庫，則為 True。
usesuper	布林值	如果使用者是超級使用者，則為 True。
usecatupd	布林值	若為 true 表示使用者可以更新系統目錄。
passwd	text	密碼。
valuntil	abstime	密碼的過期日期和時間。
useconfig	text[]	執行期變數的工作階段預設值。
useconnlimit	text	使用者可以開啟的連線數量。

查詢目錄資料表

主題

- 目錄查詢範例

通常，您可以將目錄資料表和檢視 (其名稱開頭為 **PG_** 的關係) 聯結至 Amazon Redshift 資料表和檢視。

目錄資料表使用 Amazon Redshift 不支援的各種資料類型。下列資料類型在將查詢對 Amazon Redshift 資料表聯結目錄資料表時受到支援：

- bool
- "char"
- float4
- int2
- int4
- int8
- name
- oid
- text
- varchar

如果您寫入加入查詢，其明確或隱含地參考擁有不受支援之資料類型的欄位，則查詢會傳回錯誤。在某些目錄資料表中使用的 SQL 函數也不受支援，除非是 PG_SETTINGS 和 PG_LOCKS 資料表所用的那些函數。

例如，PG_STATS 資料表不得與 Amazon Redshift 資料表一同受到查詢，因為此功能不受支援。

下列目錄資料表和檢視提供的資訊很有用，您可以將其與 Amazon Redshift 資料表中的資訊結合在一起。因為資料類型和函數限制，部分資料表僅允許部分存取權。當您查詢可部分存取的資料表時，請小心地選取或參考其欄位。

以下資料表是可完整存取的，且未包含不受支援的函數類型：

- pg_attribute
- pg_cast
- pg_depend
- pg_description

- pg_locks
- pg_opclass

以下資料表是可部分存取的，且包含一些不受支援類型、函數和遭截斷的文字欄位。文字欄位中的值遭截斷為 varchar(256) 值。

- pg_class
- pg_constraint
- pg_database
- pg_group
- pg_language
- pg_namespace
- pg_operator
- pg_proc
- pg_settings
- pg_statistic
- pg_tables
- pg_type
- pg_user
- pg_views

未在此處列出的目錄資料表為不可存取或可能未供 Amazon Redshift 管理員使用。然而，如果您的查詢不包含對 Amazon Redshift 資料表的加入時，您可以開放地查詢任何目錄資料表或檢視。

您可以使用 Postgres 目錄資料表中的 OID 欄位做為加入欄位。例如，加入條件 `pg_database.oid = stv_tbl_perm.db_id` 符合每個 PG_DATABASE 列的內部資料庫物件 ID，內含 STV_TBL_PERM 資料表中可見的 DB_ID 欄位。OID 欄位是內部主要索引鍵，當您從資料表進行選取時無法看見此欄位。目錄檢視沒有 OID 欄位。

某些 Amazon Redshift 函數必須只能在運算節點上執行。如果查詢參考的是使用者建立的資料表，SQL 會在運算節點上執行。

查詢如果只參考目錄資料表 (具有 PG 字首的資料表，例如 PG_TABLE_DEF)，或是未參考任何資料表，就只會在領導節點上執行。

如果使用運算節點函數的查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表傳回下列錯誤。

```
[Amazon](500310) Invalid operation: One or more of the used functions must be applied
 on at least one user created table.
```

下列 Amazon Redshift 函數為僅運算節點函數：

系統資訊函數

- LISTAGG
- MEDIAN
- PERCENTILE_CONT
- PERCENTILE_DISC 和 APPROXIMATE PERCENTILE_DISC

目錄查詢範例

下列查詢顯示幾個方法，您可以用這些方法來查詢目錄資料表來取得 Amazon Redshift 資料庫的相關有用資訊。

檢視資料表 ID、資料庫、結構描述和資料表名稱

下列檢視定義加入 STV_TBL_PERM 系統資料表和 PG_CLASS、PG_NAMESPACE 和 PG_DATABASE 系統目錄資料表，以傳回資料表 ID、資料庫名稱、結構描述名稱和資料表名稱。

```
create view tables_vw as
select distinct(id) table_id
,trim(datname)   db_name
,trim(nspname)   schema_name
,trim(relname)   table_name
from stv_tbl_perm
join pg_class on pg_class.oid = stv_tbl_perm.id
join pg_namespace on pg_namespace.oid = relnamespace
join pg_database on pg_database.oid = stv_tbl_perm.db_id;
```

下列範例會回傳資料表 ID 117855 的資訊。

```
select * from tables_vw where table_id = 117855;
```

```
table_id | db_name   | schema_name | table_name
```

```
---------+-----------+-------------+-----------
  117855 |       dev | public      | customer
```

根據 Amazon Redshift 資料表列出欄數

下列查詢會聯結一些目錄資料表，以瞭解每個 Amazon Redshift 資料表包含多少個欄。Amazon Redshift 資料表名稱同時儲存在 PG_TABLES 和 STV_TBL_PERM 中；請盡可能使用 PG_TABLES 傳回 Amazon Redshift 資料表名稱。

此查詢不包含任何 Amazon Redshift 資料表。

```
select nspname, relname, max(attnum) as num_cols
from pg_attribute a, pg_namespace n, pg_class c
where n.oid = c.relnamespace and  a.attrelid = c.oid
and c.relname not like '%pkey'
and n.nspname not like 'pg%'
and n.nspname not like 'information%'
group by 1, 2
order by 1, 2;

nspname | relname  | num_cols
--------+----------+----------
public  | category |        4
public  | date     |        8
public  | event    |        6
public  | listing  |        8
public  | sales    |       10
public  | users    |       18
public  | venue    |        5
(7 rows)
```

列出資料庫中的結構描述和資料表

下列查詢會將 STV_TBL_PERM 加入至部分 PG 資料表以在 TICKIT 資料庫和其結構描述名稱 (NSPNAME 欄位) 中傳回資料表清單。查詢也會傳回每個資料表中的列總數。(當系統中多個結構描述的資料表名稱相同時，此查詢相當實用。)

```
select datname, nspname, relname, sum(rows) as rows
from pg_class, pg_namespace, pg_database, stv_tbl_perm
where pg_namespace.oid = relnamespace
and pg_class.oid = stv_tbl_perm.id
and pg_database.oid = stv_tbl_perm.db_id
```

```
and datname ='tickit'
group by datname, nspname, relname
order by datname, nspname, relname;

datname | nspname | relname  |  rows
--------+---------+----------+--------
tickit  | public  | category |     11
tickit  | public  | date     |    365
tickit  | public  | event    |   8798
tickit  | public  | listing  | 192497
tickit  | public  | sales    | 172456
tickit  | public  | users    |  49990
tickit  | public  | venue    |    202
(7 rows)
```

列出資料表 ID、資料類型、欄位名稱和資料表名稱

下列查詢列出每個使用者資料表和其欄位的部分相關資訊：資料表 ID、資料表名稱、其欄位名稱和每個欄位的資料類型：

```
select distinct attrelid, rtrim(name), attname, typname
from pg_attribute a, pg_type t, stv_tbl_perm p
where t.oid=a.atttypid and a.attrelid=p.id
and a.attrelid between 100100 and 110000
and typname not in('oid','xid','tid','cid')
order by a.attrelid asc, typname, attname;

attrelid |  rtrim   |    attname     |  typname
---------+----------+----------------+-----------
  100133 | users    | likebroadway   | bool
  100133 | users    | likeclassical  | bool
  100133 | users    | likeconcerts   | bool
...
  100137 | venue    | venuestate     | bpchar
  100137 | venue    | venueid        | int2
  100137 | venue    | venueseats     | int4
  100137 | venue    | venuecity      | varchar
...
```

為資料表中的每個欄位計數資料區塊數

下列查詢會將 STV_BLOCKLIST 資料表加入至 PG_CLASS，以傳回 SALES 資料表中欄位的儲存資訊。

```
select col, count(*)
from stv_blocklist s, pg_class p
where s.tbl=p.oid and relname='sales'
group by col
order by col;

col | count
----+-------
  0 |     4
  1 |     4
  2 |     4
  3 |     4
  4 |     4
  5 |     4
  6 |     4
  7 |     4
  8 |     4
  9 |     8
 10 |     4
 12 |     4
 13 |     8
(13 rows)
```

組態參考

主題

- enable_spectrum_oid
- spectrum_query_maxerror
- statement_timeout
- stored_proc_log_min_messages
- timezone
- use_fips_ssl
- wlm_query_slot_count

修改伺服器組態

您可以採用下列方式來變更伺服器組態：

- 使用 SET 命令時，將僅覆寫目前工作階段持續時間的設定。

  例如：

  ```
  set extra_float_digits to 2;
  ```

- 修改叢集的參數群組設定。此參數群組設定包含您可設定的額外參數。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 參數群組。
- 使用 ALTER USER 命令，將指定之使用者執行的所有工作階段的組態參數設定為新的值。

  ```
  ALTER USER username SET parameter { TO | = } { value | DEFAULT }
  ```

使用 SHOW 可命令檢視目前的參數設定。使用 SHOW ALL 可以檢視您可使用 SET 命令加以設定的所有設定。

```
SHOW ALL;

name                        | setting
----------------------------+--------------
analyze_threshold_percent | 10
datestyle                   | ISO, MDY
extra_float_digits          | 2
query_group                 | default
search_path                 | $user, public
statement_timeout           | 0
```

```
timezone                    | UTC
wlm_query_slot_count        | 1
```

> **Note**
>
> 請注意，組態參數會套用至您在資料倉儲中連線的資料庫。

analyze_threshold_percent

值 (粗體為預設值)

10、0 到 100.0

描述

設定為了分析資料表而變更的資料列百分比臨界值。為了減少處理時間並提高整體系統效能，如果資料表中變更的列數百分比低於 analyze_threshold_percent 指定的值，Amazon Redshift 會略過該資料表的 ANALYZE。例如，若資料表有 100,000,000 列資料，自從上次 ANALYZE 後其中 9,000,000 列已變更，則依預設會略過此資料表，因為已變更的資料列低於百分之 10。若要在只有少量資料列變更時分析資料表，請將 analyze_threshold_percent 設定為任何更小的數。例如，如果將analyze_threshold_percent 設為 0.01，則 100,000,000 資料列的資料表中變更的資料列大於 10,000 列資料時，便不會略過資料表。若要分析所有資料表，即使沒有任何資料列變更，請將 analyze_threshold_percent 設為 0。

您只能針對目前工作階段來使用 SET 命令修改 analyze_threshold_percent 參數。不能在參數群組中修改參數。

範例

```
set analyze_threshold_percent to 15;
set analyze_threshold_percent to 0.01;
set analyze_threshold_percent to 0;
```

cast_super_null_on_error

值 (粗體為預設值)

on、off

描述

指定如果您嘗試存取物件或陣列元素的不存在成員，且您的查詢是在預設的寬鬆模式下執行，Amazon Redshift 會傳回 NULL 值。

datashare_break_glass_session_var

值 (粗體為預設值)

沒有預設值。該值可以是 Amazon Redshift 在發生不建議執行的操作時產生的任何字元字串，如下所述。

描述

套用允許某些作業的權限，這些作業通常不建議用於 AWS Data Exchange 資料查看。

在一般情況下，我們建議您不要刪除或使用刪除數據清理或改變 AWS Data Exchange 數據清理設置可公開訪問的語句更改數據存取。若要允許卸除或變更 AWS Data Exchange 資料清單以關閉可公開存取的設定，請將`datashare_break_glass_session_var`變數設定為一次性值。此一次性值由 Amazon Redshift 產生，並且會在初次嘗試有問題的操作後，在錯誤訊息中提供。

將變數設定為一次性的產生值之後，再次執行 DROP DATASHARE 或 ALTER DATASHARE 陳述式。

如需詳細資訊，請參閱 ALTER DATASHARE 使用須知 或 DROP DATASHARE 使用須知 。

範例

```
set datashare_break_glass_session_var to '620c871f890c49';
```

datestyle

值 (粗體為預設值)

格式規格 (ISO、Postgres、SQL 或 German)，以及年/月/日順序 (DMY、MDY、YMD)。

- ISO — 使用 YYYY-MM-DD HH:MM:SS 的 datestyle。
- Postgres — 使用 MM-DD HH:MM:SS YYYY 的 datestyle。
- SQL — 使用 MM-DD-YYYY HH:MM:SS 的 datestyle。
- 德文 — 使用 DD-MM-YYYY HH:MM:SS 的 datestyle。

描述

設定日期和時間值的顯示格式，以及解譯模糊日期輸入值的規則。此字串有兩個參數，可以分開或同時變更。

範例

```
show datestyle;
DateStyle
-----------
ISO, MDY
(1 row)

set datestyle to 'SQL,DMY';
```

default_geometry_encoding

值 (粗體為預設值)

描述

工作階段組態，指定此工作階段期間建立的空間幾何圖形是否使用週框方塊進行編碼。如果 `default_geometry_encoding` 是 `1`，則不會使用週框方塊對幾何圖形進行編碼。如果 `default_geometry_encoding` 是 `2`，則使用週框方塊對幾何圖形進行編碼。如需週框方塊支援的相關資訊，請參閱 邊界框。

describe_field_name_in_uppercase

值 (粗體為預設值)

off (false)、on (true)

描述

指定 SELECT 陳述式傳回的欄位名稱要使用大寫或小寫。如果此參數為開啟 (on)，會傳回大寫的資料欄名稱。如果此參數為關閉 (off)，會傳回小寫的資料欄名稱。無論 `describe_field_name_in_uppercase` 的設定為何，Amazon Redshift 都會以小寫方式儲存資料欄名稱。

範例

```
set describe_field_name_in_uppercase to on;

show describe_field_name_in_uppercase;

DESCRIBE_FIELD_NAME_IN_UPPERCASE
--------------------------------
on
```

downcase_delimited_identifier

值 (粗體為預設值)

on、off

描述

此組態即將淘汰。反之，請使用 enable_case_sensitive_identifier。

啟用超級剖析器以讀取大寫或混合大小寫的 JSON 欄位。為資料庫、結構描述、資料表和資料欄名稱使用混合大小寫的支援 PostgreSQL 資料庫啟用聯合查詢支援。若要使用區分大小寫的識別碼，請將此參數設定為 off。

使用須知

- 如果您使用的是資料列層級安全或動態資料遮罩功能，建議您在叢集或工作群組的參數群組中設定 downcase_delimited_identifier 值。這樣可確保在建立和附加政策的過程中 downcase_delimited_identifier 保持不變，然後查詢已套用政策的關係。如需有關資料列層級安全性詳細資訊，請參閱 資料列層級安全性。如需動態資料遮罩的詳細資訊，請參閱 動態資料遮罩。
- 當您將 downcase_delimited_identifier 設定為 off 並建立資料表時，您可以設定區分大小寫的資料欄名稱。當您將 downcase_delimited_identifier 設定為 on 並查詢資料表時，資料欄名稱會被變更為小寫。這可能會在 downcase_delimited_identifier 設定為 off 時產生不同的查詢結果。請思考下列範例：

```
SET downcase_delimited_identifier TO off;
--Amazon Redshift preserves case for column names and other identifiers.
```

```
--Create a table with two columns that are identical except for the case.
CREATE TABLE t ("c" int, "C" int);

INSERT INTO t VALUES (1, 2);

SELECT * FROM t;

 c | C
---+---
 1 | 2
(1 row)

SET enable_downcase_delimited_identifier TO on;
--Amazon Redshift no longer preserves case for column names and other identifiers.

SELECT * FROM t;

 c | c
---+---
 1 | 1
(1 row)
```

- 我們建議一般使用者查詢具有動態資料遮罩或資料列層級安全政策的資料表時，使用預設的 downcase_delimited_identifier 設定。如需取得資料列層級安全的相關資訊，請參閱 資料列層級安全性。如需動態資料遮罩的詳細資訊，請參閱 動態資料遮罩。

enable_case_sensitive_identifier

值 (粗體為預設值)

true、false

描述

組態值，用於決定資料庫、資料表和資料欄的名稱識別碼是否區分大小寫。以雙引號括住名稱識別碼時，識別碼的大小寫會保留。當您將 enable_case_sensitive_identifier 設定為 true 時，名稱識別碼的大小寫會保留。當您將 enable_case_sensitive_identifier 設定為 false 時，名稱識別碼的大小寫不會保留。

無論 enable_case_sensitive_identifier 組態選項的設定為何，都會保留以雙引號括住的 username 大小寫。

範例

下列範例說明如何為資料表和資料欄名稱建立及使用區分大小寫的識別碼。

```
-- To create and use case sensitive identifiers
SET enable_case_sensitive_identifier TO true;

-- Create tables and columns with case sensitive identifiers
CREATE TABLE "MixedCasedTable" ("MixedCasedColumn" int);

CREATE TABLE MixedCasedTable (MixedCasedColumn int);

-- Now query with case sensitive identifiers
SELECT "MixedCasedColumn" FROM "MixedCasedTable";

MixedCasedColumn
------------------
(0 rows)


SELECT MixedCasedColumn FROM MixedCasedTable;

mixedcasedcolumn
------------------
(0 rows)
```

下列範例說明未保留識別碼大小寫的時機。

```
-- To not use case sensitive identifiers
RESET enable_case_sensitive_identifier;

-- Mixed case identifiers are lowercased
CREATE TABLE "MixedCasedTable2" ("MixedCasedColumn" int);

CREATE TABLE MixedCasedTable2 (MixedCasedColumn int);

ERROR:  Relation "mixedcasedtable2" already exists


SELECT "MixedCasedColumn" FROM "MixedCasedTable2";

 mixedcasedcolumn
------------------
```

```
(0 rows)


SELECT MixedCasedColumn FROM MixedCasedTable2;

 mixedcasedcolumn
------------------
(0 rows)
```

使用須知

- 如果您對具體化視觀表使用自動重新整理，建議您在叢集或工作群組的參數群組中設定 enable_case_sensitive_identifier 值。這可確保在重新整理具體化視觀表時 enable_case_sensitive_identifier 保持不變。如需重新整理具體化視觀表的相關資訊，請參閱 重新整理具體化視觀表。如需在參數群組中設定組態值的相關資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 參數群組。
- 如果您使用的是資料列層級安全或動態資料遮罩功能，建議您在叢集或工作群組的參數群組中設定 enable_case_sensitive_identifier 值。這樣可確保在建立和附加政策的過程中 enable_case_sensitive_identifier 保持不變，然後查詢已套用政策的關係。如需有關資料列層級安全性詳細資訊，請參閱 資料列層級安全性。如需動態資料遮罩的詳細資訊，請參閱 動態資料遮罩。
- 當您將 enable_case_sensitive_identifier 設定為 on 並建立資料表時，您可以設定區分大小寫的資料欄名稱。當您將 enable_case_sensitive_identifier 設定為 off 並查詢資料表時，資料欄名稱會被變更為小寫。這可能會在 enable_case_sensitive_identifier 設定為 on 時產生不同的查詢結果。請思考下列範例：

```
SET enable_case_sensitive_identifier TO on;
--Amazon Redshift preserves case for column names and other identifiers.

--Create a table with two columns that are identical except for the case.
CREATE TABLE t ("c" int, "C" int);

INSERT INTO t VALUES (1, 2);

SELECT * FROM t;

 c | C
---+---
 1 | 2
(1 row)
```

```
SET enable_case_sensitive_identifier TO off;
--Amazon Redshift no longer preserves case for column names and other identifiers.

SELECT * FROM t;

 c | c
---+---
 1 | 1
(1 row)
```

- 我們建議一般使用者查詢具有動態資料遮罩或資料列層級安全政策的資料表時，使用預設的 enable_case_sensitive_identifier 設定。如需有關資料列層級安全性詳細資訊，請參閱 資料列層級安全性。如需動態資料遮罩的詳細資訊，請參閱 動態資料遮罩。

enable_case_sensitive_super_attribute

值 (粗體為預設值)

true、false

描述

組態值，決定瀏覽具有非分隔屬性名稱的 SUPER 資料類型結構是否區分大小寫。當您將 `enable_case_sensitive_super_attribute` 設定為 `true` 時，瀏覽具有非分隔屬性名稱的 SUPER 類型結構會區分大小寫。當您將值設定為 `false` 時，瀏覽具有非分隔屬性名稱的 SUPER 類型結構不會區分大小寫。

無論 `enable_case_sensitive_super_attribute` 組態選項的設定為何，以雙引號括住屬性名稱並將 `enable_case_sensitive_identifier` 設定為 `true` 時，大小寫一律會保留。

`enable_case_sensitive_super_attribute` 僅適用於具有 SUPER 資料類型的資料欄。對於所有其他資料欄，請考慮使用 `enable_case_sensitive_identifier` 代替。

如需 SUPER 資料類型的相關資訊，請參閱 SUPER 類型 和 在 Amazon Redshift 中擷取和查詢半結構化資料。

範例

下列範例說明在 `enable_case_sensitive_super_attribute` 已啟用和已停用的情況下選取 SUPER 值的結果。

```
--Create a table with a SUPER column.
CREATE TABLE tbl (col SUPER);

--Insert values.
INSERT INTO tbl VALUES (json_parse('{
 "A": "A", "a": "a"
}'));

SET enable_case_sensitive_super_attribute TO ON;

SELECT col.A FROM tbl;
  a
-----
 "A"
(1 row)

SELECT col.a FROM tbl;
  a
-----
 "a"
(1 row)

SET enable_case_sensitive_super_attribute TO OFF;

SELECT col.A FROM tbl;
  a
-----
 "a"
(1 row)

SELECT col.a FROM tbl;
  a
-----
 "a"
(1 row)
```

使用須知

- 檢視和具體化視觀表會遵循其建立時的 enable_case_sensitive_super_attribute 值。近期繫結檢視、預存程序和使用者定義函數都會遵循查詢時的 enable_case_sensitive_super_attribute 值。

- 如果您對具體化視觀表使用自動重新整理，建議您在叢集或工作群組的參數群組中設定 enable_case_sensitive_identifier value。這可確保在重新整理具體化視觀表時 enable_case_sensitive_identifier 保持不變。如需重新整理具體化視觀表的相關資訊，請參閱 重新整理具體化視觀表。如需在參數群組中設定組態值的相關資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 參數群組。
- 無論 enable_case_sensitive_super_attribute 的值為何，陳述式結果中的資料欄名稱一律會變更為小寫。若要讓資料欄名稱區分大小寫，請啟用 enable_case_sensitive_identifier。
- 我們建議一般使用者查詢具有資料列層級安全政策的資料表時，使用預設的 enable_case_sensitive_identifier 設定。如需取得資料列層級安全的相關資訊，請參閱 資料列層級安全性。

enable_numeric_rounding

值 (粗體為預設值)

on (true)、ff (false)

描述

指定是否使用數值四捨五入。如果 enable_numeric_rounding 是 on，Amazon Redshift 會在將 NUMERIC 值轉換為其他數值類型 (例如 INTEGER 或 DECIMAL 時) 使用四捨五入。如果 enable_numeric_rounding 是 off，Amazon Redshift 會在將 NUMERIC 值轉換為其他數值類型時將其截斷。如需數值類型的相關資訊，請參閱 數值類型。

範例

```
--Create a table and insert the numeric value 1.5 into it.
CREATE TABLE t (a numeric(10, 2));

INSERT INTO t VALUES (1.5);

SET enable_numeric_rounding to ON;
--Amazon Redshift now rounds NUMERIC values when casting to other numeric types.

SELECT a::int FROM t;

 a
---
```

```
 2
(1 row)


SELECT a::decimal(10, 0) FROM t;

 a
---
 2
(1 row)


 SELECT a::decimal(10, 5) FROM t;

    a
---------
 1.50000
(1 row)


SET enable_numeric_rounding to OFF;
--Amazon Redshift now truncates NUMERIC values when casting to other numeric types.

SELECT a::int FROM t;

 a
---
 1
(1 row)


SELECT a::decimal(10, 0) FROM t;

 a
---
 1
(1 row)


SELECT a::decimal(10, 5) FROM t;

    a
---------
 1.50000
```

```
(1 row)
```

enable_result_cache_for_session

值 (粗體為預設值)

on (true)、off (false)

描述

指定是否使用查詢結果快取。如果 enable_result_cache_for_session 為 on，當提交查詢時，Amazon Redshift 會檢查查詢結果的有效快取副本。如果在結果快取中找到符合者，Amazon Redshift 會使用快取的結果，不會執行查詢。如果 enable_result_cache_for_session 為 off，Amazon Redshift 會在提交查詢時忽略結果快取並執行所有查詢。

範例

```
SET enable_result_cache_for_session TO off;
--Amazon Redshift now ignores the results cache
```

enable_vacuum_boost

值 (粗體為預設值)

false、true

描述

指定是否為工作階段中所有的 VACUUM 命令執行啟用清空提升選項。如果 enable_vacuum_boost 是 true，Amazon Redshift 會在工作階段中使用 BOOST 選項執行所有 VACUUM 命令。如果 enable_vacuum_boost 是 false，則根據預設 Amazon Redshift 不會使用 BOOST 選項執行。如需 BOOST 選項的相關資訊，請參閱 VACUUM。

error_on_nondeterministic_update

值 (粗體為預設值)

false、true

描述

指定每列具有多個相符項目的 UPDATE 查詢是否會傳回錯誤。

範例

```
SET error_on_nondeterministic_update TO true;

CREATE TABLE t1(x1 int, y1 int);

CREATE TABLE t2(x2 int, y2 int);

INSERT INTO t1 VALUES (1,10), (2,20), (3,30);

INSERT INTO t2 VALUES (2,40), (2,50);

UPDATE t1 SET y1=y2 FROM t2 WHERE x1=x2;

ERROR: Found multiple matches to update the same tuple.
```

extra_float_digits

值 (粗體為預設值)

0、-15 到 2

描述

設定浮點值 (包括 float4 和 float8) 顯示的位數。此值將添加到標準位數 (適當時為 FLT_DIG 或 DBL_DIG)。該值最高可以設定為 2，以包括部分有效數字。這對於輸出必須完全還原的浮點數資料特別有用。或者，可將此值設為負數，隱匿不要的數字。

範例

下列 範例會將 `extra_float_digits` 設定為 `-2`。首先，顯示目前的參數設定。

```
show all;
      name                      | setting
--------------------------+----------------
```

```
analyze_threshold_percent | 10
datestyle                 | ISO, MDY
extra_float_digits        | 2
query_group               | default
search_path               | $user, public
statement_timeout         | 0
timezone                  | UTC
wlm_query_slot_count      | 1
```

然後，將新值設定為 -2。

```
set extra_float_digits to -2;
```

最後顯示更新的參數設置。

```
show all;
      name                    | setting
------------------------------+----------------
analyze_threshold_percent | 10
datestyle                 | ISO, MDY
extra_float_digits        | -2
query_group               | default
search_path               | $user, public
statement_timeout         | 0
timezone                  | UTC
wlm_query_slot_count      | 1
```

間隔 _ 禁止複合 _ 文字

值 (粗體為預設值)

假，真

描述

會話配置，修改包含年到月和日到第二部分的間隔值。

如果interval_forbid_composite_literals是true，如果遇到同時具有「年至月」和「日到第二」部分的間隔，則返回錯誤。例如，下面的 SQL 包含一個間隔日到第二與年到月和日到第二部分。

```
SELECT INTERVAL '1 year 1 day' DAY TO SECOND;
ERROR:  Interval Day To Second literal cannot contain year-month parts. Disable the GUC
 interval_forbid_composite_literals to suppress this error and silently discard the
 year-month part.
```

如果interval_forbid_composite_literals是false，Amazon Redshift 會抑制錯誤，並將「年到月」部分從「日」到第二個值截斷。例如，下面的 SQL 包含一個間隔日到第二與年到月和日到第二部分。

```
SET interval_forbid_composite_literals to "false";
SELECT INTERVAL '1 year 1 day' DAY TO SECOND;


 intervald2s
------------------------------
1 days 0 hours 0 mins 0.0 secs
```

json_serialization_enable

值 (粗體為預設值)

false、true

描述

修改 ORC、JSON、Ion 和 Parquet 格式資料的 JSON 序列化行為的工作階段組態。如果 json_serialization_enable 是 true，則所有頂層集合都會自動序列化為 JSON，並以 VARCHAR(65535) 的形式傳回。非複雜資料欄不會受到影響或序列化。由於集合資料欄已序列化為 VARCHAR(65535)，因此無法再直接存取其巢狀子欄位，以做為查詢語法 (也就是篩選條件子句中) 的一部分。如果 json_serialization_enable 是 false，則頂層集合不會序列化為 JSON。如需取得巢狀 JSON 序列化的相關資訊，請參閱 序列化複雜的巢狀 JSON。

json_serialization_parse_nested_strings

值 (粗體為預設值)

false、true

描述

修改 ORC、JSON、Ion 和 Parquet 格式資料的 JSON 序列化行為的工作階段組態。當 `json_serialization_parse_nested_strings` 和 `json_serialization_enable` 都為 true 時，則會剖析儲存在複雜類型 (例如對應、結構或陣列) 中的字串值，並直接以內嵌方式寫入結果中，但前提是字串值是有效的 JSON.。如果 `json_serialization_parse_nested_strings` 為 false，則巢狀複雜類型中的字串會序列化為逸出的 JSON 字串。如需詳細資訊，請參閱 序列化包含 JSON 字串的複雜類型。

max_concurrency_scaling_clusters

值 (粗體為預設值)

1、0 到 10

描述

設定當並行擴展啟用時，允許的並行擴展叢集的最大數量。來看需要更多的並行擴展，請提高此數值。降低此數值可降低並行擴展叢集的用量，以及相關的帳單費用。

並行擴展叢集的數量上限是一項可調整的配額。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 配額。

max_cursor_result_set_size

值 (粗體為預設值)

0 (預設為上限值) - 14400000 MB

描述

`max_cursor_result_set_size` 參數已不再繼續使用。如需游標結果集大小的相關資訊，請參閱游標限制條件。

mv_enable_aqmv_for_session

值 (粗體為預設值)

true、false

描述

指定 Amazon Redshift 是否可以在工作階段層級上執行具體化視觀表的自動查詢重寫。

navigate_super_null_on_error

值 (粗體為預設值)

on、off

描述

指定如果您嘗試瀏覽物件或陣列元素的不存在成員，且您的查詢是在預設的寬鬆模式下執行，Amazon Redshift 會傳回 NULL 值。

parse_super_null_on_error

值 (粗體為預設值)

off、on

描述

指定如果 Amazon Redshift 嘗試剖析物件或陣列元素的不存在成員，且您的查詢是在嚴格模式下執行，Amazon Redshift 會傳回 NULL 值。

pg_federation_repeatable_read

值 (粗體為預設值)

true、false

描述

指定 PostgreSQL 資料庫中結果的聯合查詢交易隔離層級。

- 如果 `pg_federation_repeatable_read` 為 true，則會使用 REPEATABLE READ 隔離層級語意處理聯合交易。此為預設值。

- 如果 pg_federation_repeatable_read 為 false，則會使用 READ COMMITTED 隔離層級語意處理聯合交易。

如需詳細資訊，請參閱下列內容：

- 使用 Amazon Redshift 存取聯合資料時的注意事項.
- 管理並行寫入操作.

範例

下列命令會將工作階段的 pg_federation_repeatable_read 設定為 on。show 命令會顯示已設定值的值。

```
set pg_federation_repeatable_read to on;
```

```
show pg_federation_repeatable_read;

pg_federation_repeatable_read
-----------------------------
on
```

query_group

值 (粗體為預設值)

沒有預設值；可以是任何字元字串。

描述

將使用者定義的標籤套用至在相同工作階段中執行的查詢 (相同查詢群組)。此標籤會在查詢日誌中擷取。您可以用它來限制 STL_QUERY 和 STV_INFLIGHT 資料表以及 SVL_QLOG 檢視中的結果。例如，您可以對執行的每個查詢套用不同的標籤，即可唯一識別查詢，不必查閱其 ID。

伺服器組態檔案中沒有這個參數，必須在執行期使用 SET 命令設定。雖然您可以使用很長的字元字串做為標籤，但在 STL_QUERY 資料表和 SVL_QLOG 檢視的 LABEL 資料欄中，會將標籤截斷至前 30 個字元 (在 STV_INFLIGHT 中則是 15 個字元)。

以下範例將 query_group 設為 **Monday**，然後以此標籤執行數個查詢。

```
set query_group to 'Monday';
SET
select * from category limit 1;
...
...
select query, pid, substring, elapsed, label
from svl_qlog where label ='Monday'
order by query;

query | pid  |        substring                    |  elapsed  | label
------+------+-------------------------------------+-----------+--------
789   | 6084 | select * from category limit 1;     | 65468     | Monday
790   | 6084 | select query, trim(label) from ... | 1260327   | Monday
791   | 6084 | select * from svl_qlog where ..     | 2293547   | Monday
792   | 6084 | select count(*) from bigsales;      | 108235617 | Monday
...
```

search_path

值 (粗體為預設值)

'$user', public, schema_names

以逗號分隔的現有結構描述名稱清單。如果有 '$user'，清單會取代和 SESSION_USER 相同名稱的結構描述；如果沒有，則會忽略清單。

描述

當以簡單名稱參考物件 (如資料表或函數) 且不提供結構描述元件時，指定搜尋結構描述的順序：

- 外部結構描述和外部資料表不支援搜尋路徑。外部資料表必須以外部結構描述明確限定。
- 建立物件時若沒有指定特定目標結構描述，系統會將物件置於搜尋路徑中列出的第一個結構描述。如果搜尋路徑是空的，系統會傳回錯誤。
- 當其他結構描述中有同名的物件時，會使用搜尋路徑中先找到的那一個。
- 物件若不在搜尋路徑裡的任何結構描述中，則只能使用它的完整名稱 (指定包含它的結構描述) 來參考它。
- 一定會搜尋系統目錄結構描述 pg_catalog。如果路徑中有它，會依指定順序搜尋。如果沒有，會先搜尋它再搜尋路徑中的項目。

- 一定會搜尋目前工作階段的暫時資料表結構描述 pg_temp_nnn (如果存在)。可以使用 alias pg_temp 將它明確列在路徑中。如果路徑中沒有它，則會先搜尋它 (甚至在 pg_catalog 之前)。不過，只會搜尋暫時結構描述中的關係名稱 (資料表、名稱)。不會搜尋函數名稱。

範例

下列範例會建立 ENTERPRISE 結構描述，並將 search_path 設為新的結構描述。

```
create schema enterprise;
set search_path to enterprise;
show search_path;

 search_path
-------------
 enterprise
(1 row)
```

下列範例會將 ENTERPRISE 結構描述加入預設的 search_path 中。

```
set search_path to '$user', public, enterprise;
show search_path;

         search_path
-----------------------------
 "$user", public, enterprise
(1 row)
```

下列範例會將 FRONTIER 資料表加入 ENTERPRISE 結構描述中。

```
create table enterprise.frontier (c1 int);
```

當 PUBLIC.FRONTIER 資料表是在相同資料庫中建立，且使用者沒有在查詢中指定結構描述名稱，PUBLIC.FRONTIER 的優先順序會高於 ENTERPRISE.FRONTIER。

```
create table public.frontier(c1 int);
insert into enterprise.frontier values(1);
select * from frontier;

frontier
----
(0 rows)
```

```
select * from enterprise.frontier;

c1
----
1
(1 row)
```

spectrum_enable_pseudo_columns

值 (粗體為預設值)

true、false

描述

您可以藉由將 spectrum_enable_pseudo_columns 組態參數設定為 false，以停用工作階段的虛擬資料欄建立。

範例

下列命令會停用工作階段的虛擬資料欄建立。

```
set spectrum_enable_pseudo_columns to false;
```

enable_spectrum_oid

值 (粗體為預設值)

true、false

描述

您也可以將 enable_spectrum_oid 組態參數設定為 false，僅停用 $spectrum_oid 虛擬資料欄。

範例

下列命令會透過將 enable_spectrum_oid 組態參數設定為 false 來停用 $spectrum_oid 虛擬資料欄。

```
set enable_spectrum_oid to false;
```

spectrum_query_maxerror

值 (粗體為預設值)

-1、整數

描述

您可以指定整數來指出取消查詢之前可接受的錯誤數目上限。負值會關閉最誤資料處理上限。結果會記錄在 SVL_SPECTRUM_SCAN_ERROR 中。

範例

下列範例假設 ORC 資料包含多餘字元和無效的字元。my_string 的資料欄定義會指定 3 個字元的長度。以下是此範例的範例資料：

```
my_string
---------------
abcdef
gh
ab
```

下列命令會將錯誤的數目上限設定為 1 並執行查詢。

```
set spectrum_query_maxerror to 1;
SELECT my_string FROM orc_data;
```

查詢會停止並將結果記錄到 SVL_SPECTRUM_SCAN_ERROR。

statement_timeout

值 (粗體為預設值)

0 (關閉限制)、x 毫秒

描述

停止任何超過指定毫秒數的陳述式。

statement_timeout 值是指查詢在 Amazon Redshift 將其終止前可執行的最長時間。這個時間包括規劃、排入工作負載管理 (WLM) 佇列，以及執行時間。比較這個時間和 WLM 逾時 (max_execution_time) 和 QMR (query_execution_time)，其中包含唯一的執行時間。

如果 WLM 組態中也指定了 WLM 逾時 (max_execution_time)，則會使用 statement_timeout 和 WLM 逾時 (max_execution_time) 之中較小者。如需詳細資訊，請參閱 WLM 逾時。

範例

以下查詢因為時間超過 1 毫秒，所以會逾時並被取消。

```
set statement_timeout = 1;

select * from listing where listid>5000;
ERROR:  Query (150) canceled on user's request
```

stored_proc_log_min_messages

值 (粗體為預設值)

LOG, INFO, NOTICE, WARNING, EXCEPTION

描述

指定引發預存程序訊息的最低記錄層級。系統會記錄指定層級或高於指定層級的訊息。預設值為 LOG (記錄所有訊息)。記錄層級從最高到最低的順序如下：

1. EXCEPTION
2. WARNING
3. NOTICE
4. INFO
5. LOG

例如，如果您指定 NOTICE 的值，則只會記錄 NOTICE、WARNING 和 EXCEPTION 的訊息。

timezone

值 (粗體為預設值)

UTC、時區

語法

```
SET timezone { TO | = } [ time_zone | DEFAULT ]

SET time zone [ time_zone | DEFAULT ]
```

描述

設定目前工作階段的時區。時區可以是國際標準時間 (UTC) 的偏移量或是時區名稱。

> **Note**
>
> 您不能使用叢集參數群組設定此 timezone 組態參數。只能使用 SET 命令設定目前工作階段的時區。若要設定特定資料庫使用者執行之所有工作階段的時區，請使用 ALTER USER 命令。ALTER USER ... SET TIMEZONE 會變更後續工作階段的時區，不會變更目前工作階段。

當您使用 SET timezone 命令 (time 和 zone 之間無空格) 搭配 TO 或 = 設定時區時，可指定時區名稱 time_zone、POSIX 樣式的格式偏移量、或 ISO-8601 格式偏移量，如下所示。

```
SET timezone { TO | = } time_zone
```

當您使用 SET time zone 命令設定時區而不搭配 TO 或 = 時，可以指定 time_zone (使用 INTERVAL 和時區名稱)、POSIX 樣式的格式偏移量、或 ISO-8601 格式偏移量，如下所示。

```
SET time zone time_zone
```

時區格式

Amazon Redshift 支援以下時區格式：

- 時區名稱
- INTERVAL

- POSIX 樣式的時區規格
- ISO-8601 偏移量

因為時區縮寫 (例如 PST、PDT) 已定義為 UTC 的固定偏移量，且不包括日光節約時間規則，因此 SET 命令不支援時區縮寫。

請參閱以下內容，了解時區格式的詳細資訊。

時區名稱 – 完整的時區名稱，例如 America/New_York。完整時區名稱可能包含日光節約時間規則。

以下是完整時區名稱的範例：

- Etc/Greenwich
- America/New_York
- CST6CDT
- GB

> **Note**
>
> 許多時區名稱同時也是縮寫，例如 EST、MST、NZ、UCT。

若要查看有效時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```

INTERVAL – 與 UTC 的偏移量。例如，PST 是 –8:00 或 –8 小時。

以下是 INTERVAL 時區偏移量的範例：

- -8:00
- –8 小時
- 30 分鐘

POSIX 樣式的格式 – STDoffset 或 STDoffsetDST 格式的時區規格，其中 STD 是時區縮寫，offset 是 UTC 以西的小時偏移量，DST 是選用的日光節約時區縮寫。日光節約時間假設為較給定偏移早一小時。

POSIX 樣式時區格式使用格林威治以西的正偏移，與 ISO-8601 慣例不同，此是使用格林威治以東的正偏移。

以下是 POSIX 樣式時區的範例：

- PST8
- PST8PDT
- EST5
- EST5EDT

> **Note**
>
> Amazon Redshift 不會驗證 POSIX 樣式時區規格，所以可能會將時區設為無效值。例如，即使將時區設為無效值，下列命令不會傳回錯誤。
>
> ```
> set timezone to ‘xxx36’;
> ```

ISO-8601 偏移量 – 與 UTC 的偏移量，格式為 ±[hh]:[mm]。

以下為 ISO-8601 偏移量的範例：

- -8:00
- +7:30

範例

以下範例會將目前工作階段的時區設為 New York。

```
set timezone = 'America/New_York';
```

以下範例會將目前工作階段的時區設為 UTC–8 (PST)。

```
set timezone to '-8:00';
```

以下範例使用 INTERVAL 將時區設為 PST。

```
set timezone interval '-8 hours'
```

以下範例會將目前工作階段的時區重設為系統預設時區 (UTC)。

```
set timezone to default;
```

若要設定資料庫使用者的時區，請使用 ALTER USER … SET 陳述式。以下範例會將 dbuser 的時區設為 New York。新值將在使用者的後續工作階段中延用。

```
ALTER USER dbuser SET timezone to 'America/New_York';
```

use_fips_ssl

值 (粗體為預設值)

true、false

描述

指定是否使用 FIPS 相容 SSL 模式的參數群組值。如果use_fips_ssl是true，則會使用符合 FIPS 標準的 SSL 模式。如果use_fips_ssl是false，則不會使用符合 FIPS 標準的 SSL 模式。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的設定連線的安全選項。

若要為 Amazon Redshift 佈建的叢集設定參數，請參閱 Amazon Redshift 管理指南中的關於參數群組。若要設定 Redshift 無伺服器的參數，請參閱亞馬遜 Redshift 管理指南中的設定符合 FIPS 標準的 SSL 連線，以及CreateWorkgrouphttps://docs.aws.amazon.com/redshift-serverless/latest/APIReference/API_CreateWorkgroup.html或UpdateWorkgrouphttps://docs.aws.amazon.com/redshift-serverless/latest/APIReference/API_UpdateWorkgroup.html在 Redshift 無伺服器 API 參考中設定與亞馬遜 Redshift 無伺服器的 SSL 連線。

wlm_query_slot_count

值 (粗體為預設值)

1、1 到 50 (不得超過服務類別可使用的槽數量 (並行層級))

描述

設定查詢所使用的查詢槽數量。

工作負載管理 (WLM) 會根據為佇列設定的並行層級，在服務類別中預留插槽。例如，如果並行層級設定為 5，則服務類別會有 5 個插槽。WLM 會為服務類別將可用記憶體平等地分配到每個槽。如需詳細資訊，請參閱 實作工作負載管理。

> **Note**
>
> 如果 wlm_query_slot_count 的值大於服務類別可使用的槽數量 (並行層級)，這時查詢會失敗。如果發生錯誤，請將 wlm_query_slot_count 減為允許的值。

對於效能受配置記憶體嚴重影響的操作，例如清空，增加 wlm_query_slot_count 的值可以提高效能。特別是執行速度慢的清空命令，請檢查 SVV_VACUUM_SUMMARY 檢視中的相應記錄。如果 SVV_VACUUM_SUMMARY 檢視中 sort_partitions 和 merge_increments 的值較大 (接近或大於 100)，下次對該資料表執行真空時請考慮增加 wlm_query_slot_count 的值。

增加 wlm_query_slot_count 值會限制可執行的並行查詢數量。例如，假設服務類別的並行層級為 5，且 wlm_query_slot_count 設為 3。在 wlm_query_slot_count 為 3 的工作階段中執行查詢時，最多可以在相同服務類別中再執行 2 個並行的查詢。之後的查詢會在佇列中等待，直到執行中的查詢完成且槽空出來。

範例

使用 SET 命令設定目前工作階段持續時間的 wlm_query_slot_count 值。

```
set wlm_query_slot_count to 3;
```

文件歷史記錄

> **Note**
>
> 如需 Amazon Redshift 中新功能的說明，請參閱新增功能。

下表說明在 2018 年 5 月之後對 Amazon Redshift 資料庫開發人員指南進行的重要文件變更。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

API 版本：2012-12-01

如需《Amazon Redshift 管理指南》的變更清單，請參閱 Amazon Redshift 管理指南文件歷史記錄。

如需新功能的詳細資訊，包括每個版本的修正清單以及關聯的叢集版本編號，請參閱叢集版本歷史記錄。

變更	描述	日期
支援空間 3D 和 4D 幾何圖形以及新的空間函數	您現在可以使用額外的空間函數，且 3D 和 4D 幾何圖形支援已新增到某些函數中。	2021 年 8 月 19 日
支援資料欄壓縮編碼以實現自動資料表最佳化	您可以為資料表指定 ENCODE AUTO 選項，以自動管理資料表中所有資料欄的壓縮編碼。	2021 年 8 月 3 日
支援使用 Amazon Redshift 資料 API 搭配多個或一個具有參數的 SQL 陳述式	您現在可以搭配 Amazon Redshift 資料 API 執行多個或一個具有參數的 SQL 陳述式。	2021 年 7 月 28 日
支援搭配資料欄層級覆寫使用不區分大小寫定序	您現在可以在 CREATE DATABASE 陳述式中使用 COLLATE 子句來指定預設定序。	2021 年 6 月 24 日
支援跨帳戶共用資料	您現在可以在 AWS 帳戶 之間共用資料。	2021 年 4 月 30 日

支援使用遞迴 CTE 的階層式資料查詢	您現在可以在 SQL 中使用遞迴通用資料表運算式 (CTE)。	2021 年 4 月 29 日
支援跨資料庫查詢	您現在可以在叢集中的各個資料庫上查詢資料。	2021 年 3 月 10 日
支援在 COPY 和 UNLOAD 命令上進行精細定義存取控制	您現在可以授予執行 COPY 和 UNLOAD 命令的權限給 Amazon Redshift 叢集中的特定使用者和群組，以建立更精細的存取控制政策。	2021 年 1 月 12 日
支援原生 JSON 和半結構化資料	您現在可以定義 SUPER 資料類型。	2020 年 12 月 9 日
支援 MySQL 的聯合查詢	您現在可以將聯合查詢寫入支援的 MySQL 引擎。	2020 年 12 月 9 日
支援資料共用	您可以在 Amazon Redshift 叢集之間共用資料。	2020 年 12 月 9 日
支援自動資料表最佳化	您現在可以定義自動的分佈和排序索引鍵。	2020 年 12 月 9 日
支援 Amazon Redshift ML	您現在可以建立、訓練和部署機器學習 (ML) 模型。	2020 年 12 月 8 日
支援具體化視觀表的自動重新整理和查詢重寫	您現在可以保留具 up-to-date 有自動重新整理的具體化視觀表，而且可透過自動重新寫入來改善查詢	2020 年 11 月 11 日
支援 TIME 和 TIMETZ 資料類型	您現在可以使用 TIME 和 TIMETZ 資料類型建立資料表。TIME 資料類型會儲存一天中的時間但不包含時區資訊，而 TIMETZ 會儲存一天中的時間，且包括時區資訊	2020 年 11 月 11 日

支援 Lambda UDF 和記號化	您現在可以撰寫 Lambda UDF 來啟用資料的外部字符化。	2020 年 10 月 26 日
支援修改資料表的資料欄編碼	您現在可以修改資料表的資料欄編碼。	2020 年 10 月 20 日
支援跨資料庫查詢	Amazon Redshift 現在可查詢叢集中的各個資料庫。	2020 年 10 月 15 日
HyperLogLog 草圖 Support	Amazon Redshift 現在可以存儲和處理 HyperLogLogSketches。	2020 年 10 月 2 日
支援 Apache Hudi 和 Delta Lake	建立 Redshift Spectrum 外部資料表的增強功能。	2020 年 9 月 24 日
支援查詢空間資料的增強功能	增強功能包括載入 Shapefile 和數個新的空間 SQL 函數。	2020 年 9 月 15 日
具體化視觀表支援外部資料表	您可以在 Amazon Redshift 中建立參考外部資料來源的具體化視觀表。	2020 年 6 月 19 日
支援寫入外部資料表	您可以透過執行 CREATE EXTERNAL TABLE AS SELECT 來寫入一個新的外部資料表或以 INSERT INTO 來插入資料到現有的外部資料表。	2020 年 6 月 8 日
支援結構描述的儲存控制	管理結構描述之儲存控制項的命令和檢視更新。	2020 年 6 月 2 日
支援聯合查詢一般可用性	更新有關使用聯合查詢來查詢資料的資訊。	2020 年 4 月 16 日
支援其他空間函數	新增其他空間函數的描述。	2020 年 4 月 2 日

支援具體化檢視的一般可用性	從叢集版本 1.0.13059 開始全面推出具體化檢視。	2020 年 2 月 19 日
支援資料欄層級權限	從叢集版本 1.0.13059 開始，即可使用資料欄層級的權限。	2020 年 2 月 19 日
ALTER TABLE	您可以使用 ALTER TABLE 命令搭配 ALTER DISTSTYLE ALL 子句來變更資料表的分佈樣式。	2020 年 2 月 11 日
聯合查詢支援	更新指南，使用更新的 CREATE EXTERNAL SCHEMA 說明聯合查詢。	2019 年 12 月 3 日
資料湖匯出支援	更新指南，說明 UNLOAD 命令的新參數。	2019 年 12 月 3 日
空間資料支援	更新指南以說明空間資料的支援。	2019 年 11 月 21 日
新主控台支援	更新指南，描述新的 Amazon Redshift 主控台。	2019 年 11 月 11 日
自動資料表排序支援	Amazon Redshift 可以自動排序資料表資料。	2019 年 11 月 7 日
VACUUM BOOST 選項支援	您可以在清空資料表時使用 BOOST 選項。	2019 年 11 月 7 日
支援預設 IDENTITY 資料欄	您可以利用預設 IDENTITY 資料欄建立資料表。	2019 年 9 月 19 日
支援 AZ64 壓縮編碼	您可以利用 AZ64 壓縮編碼來編碼一些資料欄。	2019 年 9 月 19 日
支援查詢優先順序	您可以設定自動 WLM 佇列的查詢優先順序。	2019 年 8 月 22 日

支援AWS Lake Formation	您可以將 Lake Formation Data Catalog 與 Amazon Redshift Spectrum 搭配使用。	2019 年 8 月 8 日
COMPUPDATE PRESET	您可以將 COPY 命令搭配 COMPUPDATE PRESET 使用，以啟用 Amazon Redshift 來選擇壓縮編碼。	2019 年 6 月 13 日
ALTER COLUMN	您可以使用 ALTER TABLE 命令搭配 ALTER COLUMN 以增加 VARCHAR 資料欄的大小。	2019 年 5 月 22 日
支援預存程序	您可以在 Amazon Redshift 中定義 PL/pgSQL 預存程序。	2019 年 4 月 24 日
支援自動工作負載管理 (WLM) 組態	您可以讓 Amazon Redshift 搭配自動 WLM 一起執行。	2019 年 4 月 24 日
UNLOAD (卸載) 至 Zstandard	您可以使用 UNLOAD 命令，將 Zstandard 壓縮套用至卸載到 Amazon S3 的文字檔和逗號分隔值 (CSV) 檔案。	2019 年 4 月 3 日
並行擴展	當並行擴展啟用時，當您需要更多叢集容量以執行增加的並行讀取查詢時，Amazon Redshift 將會自動新增額外的叢集容量。	2019 年 3 月 21 日
UNLOAD 至 CSV	您可以使用 UNLOAD 命令來卸載至以 CSV 文字格式化的檔案。	2019 年 3 月 13 日

AUTO 分佈樣式	若要啟用自動分佈，您可以使用 CREATE TABLE 陳述式來指定 AUTO 分佈樣式。啟用自動分佈時，Amazon Redshift 會根據資料表資料來指派最佳分佈樣式。分佈是在幾秒內於背景中發生變化。	2019 年 1 月 23 日
來自 Parquet 的 COPY 支援 SMALLINT	COPY 現支援從 Parquet 格式的檔案載入至使用 SMALLINT 資料類型的欄。如需詳細資訊，請參閱從單欄資料格式 COPY	2019 年 1 月 2 日
DROP EXTERNAL DATABASE	您可以在 DROP SCHEMA 命令中包含 DROP EXTERNAL DATABASE 子句，以捨棄外部資料庫。	2018 年 12 月 3 日
跨區域 UNLOAD	您可以指定 REGION 參數，以 UNLOAD 到另一個 AWS 區域中的 Amazon S3 儲存貯體。	2018 年 10 月 31 日
自動清空刪除	Amazon Redshift 會在背景執行 VACUUM DELETE 操作，因此您很少需要執行 DELETE ONLY 清空。Amazon Redshift 會將 VACUUM DELETE 排程在負載降低的期間執行，並在高負載期間暫停操作。	2018 年 10 月 31 日
自動分佈	當您在 CREATE TABLE 陳述式中未指定分佈樣式時，Amazon Redshift 會根據資料表資料來指派最佳分佈樣式。分佈是在幾秒內於背景中發生變化。	2018 年 10 月 31 日

AWS Glue Data Catalog 的精細存取控制	您現在可以對 AWS Glue Data Catalog 中存放的資料指定存取等級。	2018 年 10 月 15 日
UNLOAD 與資料類型	您可以在 UNLOAD 命令中指定 MANIFEST VERBOSE 選項，將中繼資料新增至資訊清單檔案，包括欄的名稱和資料類型、檔案大小及列計數。	2018 年 10 月 10 日
使用單一 ALTER TABLE 陳述式新增多個分割區	對於 Redshift Spectrum 外部資料表，您可以在單一 ALTER TABLE ADD 陳述式中合併多個 PARTITION 子句。如需詳細資訊，請參閱更改外部資料表範例。	2018 年 10 月 10 日
UNLOAD 與標頭	您可以在 UNLOAD 命令中指定 HEADER 選項，在每個輸出檔案的頂端新增含有欄名稱的標頭行。	2018 年 9 月 19 日
新的系統資料表和檢視	新增 SVL_S3Retries、SVL_USER_INFO 和 STL_DISK_FULL_DIAG 文件。	2018 年 8 月 31 日
在 Amazon Redshift Spectrum 中支援巢狀資料	您現在可以查詢 Amazon Redshift Spectrum 資料表中存放的巢狀資料。如需詳細資訊，請參閱教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料。	2018 年 8 月 8 日

預設啟用 SQA	所有新叢集現在預設為啟用短期查詢加速 (SQA)。SQA 使用機器學習提升效能、更快產生結果，以及更準確預測查詢執行時間。如需詳細資訊，請參閱短期查詢加速。	2018 年 8 月 8 日
Amazon Redshift Advisor	針對如何提高叢集效能和降低運營成本，您現在可以從 Amazon Redshift Advisor 獲得量身訂做的建議。如需詳細資訊，請參閱 Amazon Redshift Advisor。	2018 年 7 月 26 日
立即參考別名	現在，您可以在定義別名表達式後立即參考它。如需詳細資訊，請參閱 SELECT 清單。	2018 年 7 月 18 日
建立外部資料表時指定壓縮類型	現在，您可以在使用 Amazon Redshift Spectrum 建立外部資料表時指定壓縮類型。如需詳細資訊，請參閱建立外部資料表。	2018 年 6 月 27 日
PG_LAST_UNLOAD_ID	新增有關新的系統資訊函數 PG_LAST_UNLOAD_ID 的文件。如需詳細資訊，請參閱 PG_LAST_UNLOAD_ID。	2018 年 6 月 27 日
ALTER TABLE RENAME COLUMN	ALTER TABLE 現在支援重新命名外部資料表的資料欄。如需詳細資訊，請參閱更改外部資料表範例。	2018 年 6 月 7 日

舊版更新

下表會說明 2018 年 6 月之前，《Amazon Redshift 資料庫開發人員指南》每個版本的重要變更。

變更	描述	變更日期
來自 Parquet 的 COPY 包含 SMALLINT	COPY 現支援從 Parquet 格式的檔案載入至使用 SMALLINT 資料類型的欄。如需更多資訊，請參閱 從單欄資料格式 COPY	2019 年 1 月 2 日
從資料欄格式 COPY	COPY 現在支援從 Amazon S3 上使用 Parquet 和 ORC 資料欄資料格式的檔案載入資料。如需更多資訊，請參閱 從單欄資料格式 COPY	2018 年 5 月 17 日
動態的 SQA 執行時間上限	根據預設，工作負載管理 (WLM) 現在會根據叢集工作負載分析，動態地指派短期查詢加速 (SQA) 的執行時間上限值。如需詳細資訊，請參閱短期查詢最長執行時間。	2018 年 5 月 17 日
STL_LOAD_COMMITS 中的新資料欄	STL_LOAD_COMMITS 系統資料表有一個新資料欄：`file_format` 。	2018 年 5 月 10 日
STL_HASHJOIN 和其他系統日誌資料表中的新資料欄	STL_HASHJOIN 系統資料表有三個新資料欄：`hash_segment` 、`hash_step` 、`checksum`。另外，TL_MERGEJOIN、STL_NESTLOOP、STL_HASH、STL_SCAN、STL_SORT、STL_LIMIT、STL_PROJECT 中皆已新增 `checksum`。	2018 年 5 月 17 日
STL_AGGR 中的新資料欄	STL_AGGR 系統資料表有兩個新資料欄：`resizes` 和 `flushable` 。	2018 年 4 月 19 日
REGEX 函數的新選項	現在，使用 REGEXP_INSTR 和 REGEXP_SUBSTR 函數時，您可以指定要使用出現的哪一個相符項目，以及是否執行區分大小寫的比對。REGEXP_INSTR 也可讓您指定要傳回相符項目第一個字元的位置，或是相符項目後第一個字元的位置。	2018 年 3 月 22 日
系統資料表中的新資料欄	STL_COMMIT_STATS 系統資料表中新增 tombstone dblocks、tossedblocks、batched_by 資料欄。STV_SLICES 系統檢視中新增 localslice 資料欄。	2018 年 3 月 22 日

變更	描述	變更日期
新增和丟棄外部資料表中的資料欄	對於 Amazon Redshift Spectrum 外部資料表，ALTER TABLE 現在支援 ADD COLUMN 和 DROP COLUMN。	2018 年 3 月 22 日
Redshift Spectrum 新 AWS 區域	Redshift Spectrum 現已在孟買和聖保羅區域提供。如需支援的區域的清單，請參閱 Amazon Redshift Spectrum 區域。	2018 年 3 月 22 日
資料表限制增加至 20,000	現在，8xlarge 叢集節點類型的資料表數目上限是 20,000。大型與特大型節點類型上限則是 9,900。如需詳細資訊，請參閱限制和配額。	2018 年 3 月 13 日
Redshift Spectrum 支援 JSON 和 Ion	使用 Redshift Spectrum，您可以參考包含 JSON 或 Ion 資料格式純量資料的檔案。如需詳細資訊，請參閱CREATE EXTERNAL TABLE。	2018 年 2 月 26 日
Redshift Spectrum 的 IAM 角色變更	您可以鏈結 AWS Identity and Access Management (IAM) 角色，讓您的叢集能夠擔任未連接到叢集的其他角色，包括屬於其他 AWS 帳戶的角色。如需詳細資訊，請參閱在 Amazon Redshift Spectrum 中鏈結 IAM 角色。	2018 年 2 月 1 日
ADD PARTITION 支援 IF NOT EXISTS	ALTER TABLE 的 ADD PARTITION 子句現在支援 IF NOT EXISTS 選項。如需詳細資訊，請參閱ALTER TABLE。	2018 年 1 月 11 日
外部資料表的 DATE 資料	Redshift Spectrum 外部資料表現在支援 DATE 日期類型。如需詳細資訊，請參閱CREATE EXTERNAL TABLE。	2018 年 1 月 11 日
Redshift Spectrum 新 AWS 區域	Redshift Spectrum 現已在新加坡、雪梨、首爾、法蘭克福區域提供。如需支援的 AWS 區域清單，請參閱 Amazon Redshift Spectrum 區域。	2017 年 11 月 16 日

變更	描述	變更日期
Amazon Redshift 工作負載管理 (WLM) 中的短期查詢加速	短期查詢加速 (SQA) 可排定讓短期執行的查詢優先於長期執行的查詢。SQA 會在專用佇列中執行短期查詢，所以 SQA 查詢不會被迫在佇列中排在長期查詢後面等待。SQA 可讓短期執行的查詢更快開始執行，使用者會更快看到結果。如需詳細資訊，請參閱使用短期查詢加速。	2017 年 11 月 16 日
WLM 重新指派跳轉查詢	現在，Amazon Redshift 工作負載管理 (WLM) 不再取消並重新啟動跳轉查詢，而是改成將合格查詢重新指派至新佇列。當 WLM 重新指派查詢時，會將查詢移到新佇列並繼續執行，這可節省時間和系統資源。不符合重新指派資格的跳轉查詢，則會重新啟動或取消。如需詳細資訊，請參閱WLM 查詢佇列跳轉。	2017 年 11 月 16 日
使用者存取系統日誌	根據預設，在大部份使用者可查看的系統日誌資料表中，一般使用者無法查看另一位使用者產生的資料列。若要允許一般使用者查看使用者可查看資料表中所有資料列，包括另一位使用者產生的資料列，請執行 ALTER USER 或 CREATE USER，然後將 SYSLOG ACCESS 參數設定為 UNRESTRICTED。	2017 年 11 月 16 日
結果快取	有了 結果快取，當您執行查詢時，Amazon Redshift 會快取結果。當您再次執行查詢，Amazon Redshift 會檢查有沒有該查詢結果的有效快取副本。如果在結果快取中找到符合者，Amazon Redshift 會使用快取的結果，不會執行查詢。結果快取預設為開啟。若要關閉結果快取，請將 enable_result_cache_for_session 組態參數設為 `off`。	2017 年 11 月 16 日
資料欄中繼資料函數	PG_GET_COLS 和 PG_GET_LATE_BINDING_VIEW_COLS 會傳回 Amazon Redshift 資料表、檢視、晚期繫結檢視的欄中繼資料。	2017 年 11 月 16 日

變更	描述	變更日期
CTAS 的 WLM 佇列跳轉	對於 CREATE TABLE AS (CTAS) 陳述式及唯讀查詢，例如 SELECT 陳述式，Amazon Redshift 工作負載管理 (WLM) 現在支援查詢佇列跳轉。如需詳細資訊，請參閱WLM 查詢佇列跳轉。	2017 年 10 月 19 日
Amazon Redshift Spectrum 清單檔案	當您建立 Redshift Spectrum 外部資料表時，可以指定一個其中已列出 Amazon S3 上資料檔案位置的資訊清單檔案。如需詳細資訊，請參閱CREATE EXTERNAL TABLE。	2017 年 10 月 19 日
Amazon Redshift Spectrum 新 AWS 區域	Redshift Spectrum 現已在歐洲 (愛爾蘭) 和亞太區域 (東京) 區域提供。如需支援的 AWS 區域清單，請參閱 Amazon Redshift Spectrum 考量事項。	2017 年 10 月 19 日
Amazon Redshift Spectrum 已新增檔案格式	現在，您可以建立 Regex、OpenCSV、Avro 資料檔案格式的 Amazon Redshift Spectrum 外部資料表。如需詳細資訊，請參閱CREATE EXTERNAL TABLE。	2017 年 10 月 5 日
Amazon Redshift Spectrum 外部資料表的虛擬資料欄	您可以在 Redshift Spectrum 外部資料表中選取 `$path` 和 `$size` 虛擬欄，以檢視 Amazon S3 中的參考資料檔案的位置和大小。如需詳細資訊，請參閱虛擬資料欄。	2017 年 10 月 5 日
驗證 JSON 的函數	您可以使用 IS_VALID_JSON 和 IS_VALID_JSON_ARRAY 函數來檢查 JSON 格式有效與否。其他 JSON 函數現在有選用的 `null_if_invalid` 引數。	2017 年 10 月 5 日
LISTAGG DISTINCT	您可以使用 DISTINCT 子句搭配 LISTAGG 彙總函數和 LISTAGG 視窗函數，在將值串連成單一字串之前，從指定的表達式中消除重複的值。	2017 年 10 月 5 日
以大寫名稱檢視資料欄	若要在 SELECT 結果中以大寫名稱檢視資料欄，您可以將 describe_field_name_in_uppercase 組態參數設為 `true`。	2017 年 10 月 5 日

變更	描述	變更日期
略過外部資料表的標題行	您可以在 `skip.header.line.count` 命令中設定 CREATE EXTERNAL TABLE 屬性，來略過 Redshift Spectrum 資料檔案一開始的標題行。	2017 年 10 月 5 日
掃描的資料列數	WLM 查詢監控規則使用 scan_row_count 指標來傳回掃描步驟中的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。如需詳細資訊，請參閱已佈建 Amazon Redshift 的查詢監控指標。	2017 年 9 月 21 日
SQL 使用者定義的函數	SQL 的純量使用者定義函數 (UDF) 運用了 SQL SELECT 子句，後者會在函數被用時執行，並傳回單一值。如需詳細資訊，請參閱建立純量 SQL UDF。	2017 年 8 月 31 日
近期繫結檢視	近期繫結的檢視不會與底層資料庫物件繫結，像是資料表、使用者定義的函數。因此，檢視和其參考的物件之間並無相依性。即使參考的物件不存在，您仍可以建立檢視。由於沒有相依性，您可以捨棄或修改參考物件，而不會影響檢視。Amazon Redshift 在查詢檢視之前不會檢查相依姓。若要建立近期繫結的檢視，在您的 CREATE VIEW 陳述式中指定 WITH NO SCHEMA BINDING 子句。如需詳細資訊，請參閱CREATE VIEW。	2017 年 8 月 31 日
OCTET_LENGTH 函數	OCTET_LENGTH 傳回指定字串的長度 (以位元組為單位)。	2017 年 8 月 18 日
支援 ORC 和 Grok 檔案類型	Amazon Redshift Spectrum 現在可以支援 Redshift Spectrum 資料檔案的 ORC 和 Grok 資料格式。如需詳細資訊，請參閱在 Amazon Redshift Spectrum 為查詢建立資料檔案。	2017 年 8 月 18 日
RegexSerDe 現在支援	Amazon Redshift Spectrum 現在支持數 RegexSerDe 據格式。如需詳細資訊，請參閱在 Amazon Redshift Spectrum 為查詢建立資料檔案。	2017 年 7 月 19 日

變更	描述	變更日期
SVV_TABLES 和 SVV_COLUMNS 新增新資料欄	domain_name 新增 remarks 和 SVV_COLUMNS 資料欄。SVV_TABLES 新增 remarks 資料欄。	2017 年 7 月 19 日
SVV_TABLES 和 SVV_COLUMNS 系統檢視	SVV_TABLES 和 SVV_COLUMNS 系統檢視提供本機、外部資料表與檢視資料欄相關的資訊，以及其他詳細資訊。	2017 年 7 月 7 日
使用 Amazon EMR Hive 中繼存放區的 Amazon Redshift Spectrum 不再需要 VPC	使用 Amazon EMR Hive 中繼存放區時，Redshift Spectrum 已不要求 Amazon Redshift 叢集和 Amazon EMR 叢集必須位於同一個 VPC 和同一子網路。如需詳細資訊，請參閱在 Amazon Redshift Spectrum 中使用外部目錄。	2017 年 7 月 7 日
UNLOAD 為更小的檔案	根據預設，UNLOAD 會在 Amazon S3 上建立多個檔案，檔案大小最大為 6.2 GB。若要建立更小的檔案，可在 UNLOAD 命令中指定 MAXFILESIZE。您可以指定 5 MB 至 6.2 GB 的檔案大小上限。如需詳細資訊，請參閱UNLOAD。	2017 年 7 月 7 日
TABLE PROPERTIES	您現在可以為 CREATE EXTERNAL TABLE 或 ALTER TABLE 設定 TABLE PROPERTIES numRows 參數，這會更新資料表統計以反映資料表中的資料列數。	2017 年 6 月 6 日
ANALYZE PREDICATE COLUMNS	為了節省時間和叢集資源，您可以選擇僅分析可能用來做為述詞的資料列。執行 ANALYZE 搭配 PREDICATE COLUMNS 子句時，分析操作只會含括已用在連接、篩選條件、GROUP BY 子句、或者用作排序索引鍵或分佈索引鍵的資料欄。如需詳細資訊，請參閱分析資料表。	2017 年 5 月 25 日
Amazon Redshift Spectrum 的 IAM 政策	若要授予只使用 Redshift Spectrum 的 Amazon S3 儲存貯體存取權，您可以加入允許使用者代理程式 AWS Redshift/Spectrum 存取的條件。如需詳細資訊，請參閱Amazon Redshift Spectrum 的 IAM 政策。	2017 年 5 月 25 日

變更	描述	變更日期
Amazon Redshift Spectrum 遞迴掃描	Redshift Spectrum 現在會掃描 Amazon S3 中指定資料夾以及子資料夾裡的檔案。如需詳細資訊，請參閱建立 Redshift Spectrum 外部資料表。	2017 年 5 月 25 日
查詢監控規則	使用 WLM 查詢監控規則，您可以為 WLM 查詢定義以指標為基礎的效能邊界，並指定在查詢超出那些邊界時 (日誌記錄、跳轉或中止)，要採取何種動作。請將查詢監控規則定義為工作負載管理 (WLM) 組態的一部分。如需詳細資訊，請參閱WLM 查詢監控規則。	2017 年 4 月 21 日
Amazon Redshift Spectrum	使用 Redshift Spectrum，可以有效率地查詢 Amazon S3 裡的檔案並擷取資料，無需將資料載入資料表。Redshift Spectrum 對於大型資料庫的查詢執行十分快速，因為 Redshift Spectrum 會直接掃描 Amazon S3 裡的檔案。大部分的處理發生在 Amazon Redshift Spectrum 層，大部分資料則保留在 Amazon S3。可以多個叢集同時查詢 Amazon S3 上的相同資料集，無需為每個叢集複製資料副本。如需更多資訊，請參閱 使用 Amazon Redshift Spectrum 查詢外部資料	2017 年 4 月 19 日
新系統資料表支援 Redshift Spectrum	新增下列新的系統檢視來支援 Redshift Spectrum： • SVL_S3QUERY • SVL_S3QUERY_SUMMARY • SVV_EXTERNAL_COLUMNS • SVV_EXTERNAL_DATABASES • SVV_EXTERNAL_PARTITIONS • SVV_EXTERNAL_TABLES • PG_EXTERNAL_SCHEMA	2017 年 4 月 19 日

變更	描述	變更日期
APPROXIMATE PERCENTILE_DISC 彙總函數	現已可使用 APPROXIMATE PERCENTILE_DISC 彙總函數。	2017 年 4 月 4 日
使用 KMS 的伺服器端加密	您現在可以搭配 AWS Key Management Service 金鑰 (SSE-KMS) 來使用伺服器端加密，將資料卸載到 Amazon S3。此外，COPY 現在會明顯地從 Amazon S3 載入 KMS 加密的資料檔案。如需詳細資訊，請參閱 UNLOAD。	2017 年 2 月 9 日
新的授權語法	您現在可以使用 IAM_ROLE、MASTER_SYMMETRIC_KEY、ACCESS_KEY_ID、SECRET_ACCESS_KEY、SESSION_TOKEN 參數將授權和存取資訊提供給 COPY、UNLOAD、CREATE LIBRARY 命令。新的授權語法是提供單一字串引數給 CREDENTIALS 參數的彈性替代做法。如需詳細資訊，請參閱授權參數。	2017 年 2 月 9 日
提升結構描述限制	現在每個叢集最多可以建立 9,900 個結構描述。如需詳細資訊，請參閱CREATE SCHEMA。	2017 年 2 月 9 日
預設資料表編碼	CREATE TABLE 和 ALTER TABLE 現在會為大部分新資料欄指派 LZO 壓縮編碼。定義為排序索引鍵的資料欄、定義為 BOOLEAN、REAL 或 DOUBLE PRECISION 資料類型的資料欄、以及暫時資料表，則預設指派 RAW 編碼。如需詳細資訊，請參閱ENCODE。	2017 年 2 月 6 日
ZSTD 壓縮編碼	Amazon Redshift 現在支援 ZSTD 資料欄壓縮編碼。	2017 年 1 月 19 日
PERCENTILE_CONT 和 MEDIAN 彙總函數	PERCENTILE_CONT 和 MEDIAN 現已是彙總函數也是視窗函數。	2017 年 1 月 19 日

變更	描述	變更日期
使用者定義的函數 (UDF) 的使用者日誌	您可以使用 Python 記錄模組，在 UDF 中建立使用者定義的錯誤和警告訊息。您可以在查詢執行之後，查詢 SVL_UDF_LOG 系統檢視來擷取日誌記錄的訊息。如需使用者定義的資訊詳細資訊，請參閱 記錄 UDF 中的錯誤和警告	2016 年 12 月 8 日
ANALYZE COMPRESSION 預估減少	ANALYZE COMPRESSION 命令現在支援估算每個資料欄的磁碟空間減少百分比。如需詳細資訊，請參閱ANALYZE COMPRESSION。	2016 年 11 月 10 日
連線限制	您現在可以設定允許使用者同時打開的資料庫連線數目限制。您也可以限制資料庫的並行連線數量。如需詳細資訊，請參閱 CREATE USER 及 CREATE DATABASE。	2016 年 11 月 10 日
強化 COPY 的排序	現在，當您以排序索引鍵的順序載入資料時，COPY 會自動在資料表的已排序區域中加入一個資料列。如需啟用此強化的特定需求，請參閱以排序索引鍵順序載入資料	2016 年 11 月 10 日
CTAS 的壓縮	現在，CREATE TABLE AS (CTAS) 會根據資料欄的資料類型，自動指派壓縮編碼給新資料表。如需詳細資訊，請參閱欄位和資料表屬性的繼承。	2016 年 10 月 28 日
時間戳記包含時區資料類型	對於時區 (TIMESTAMPTZ) 資料類型，Amazon Redshift 現在支援時間戳記。同時，也已新增多個新函數來支援新的資料類型。如需詳細資訊，請參閱日期和時間函數。	2016 年 9 月 29 日
分析閾值	為了減少處理時間並提高 ANALYZE 操作的整體系統效能，如果自上次執行 ANALYZE 命令以來變更後的資料欄百分比低於 analyze_threshold_percent 參數指定的分析閾值，Amazon Redshift 會跳過資料表分析。 `analyze_threshold_percent` 預設為 10。	2016 年 8 月 9 日

變更	描述	變更日期
新的 STL_RESTARTED_SESSIONS 系統資料表	當 Amazon Redshift 重新啟動工作階段時，STL_RESTARTED_SESSIONS 會記錄新的處理程序 ID (PID) 和舊的 PID。	2016 年 8 月 9 日
已更新日期和時間函數的文件	新增函數的摘要資訊和 日期和時間函數 的連結，並已更新函數的參考資訊使資訊一致。	2016 年 6 月 24 日
STL_CONNECTION_LOG 中的新資料欄	STL_CONNECTION_LOG 系統資料表有兩個新的資料欄可追蹤 SSL 連線。如果您定期將稽核日誌載入 Amazon Redshift 資料表，則需要將以下新欄位新增到目標資料表：sslcompression 和 sslexpansion。	2016 年 5 月 5 日
MD5 雜湊密碼	您可以提供密碼和使用者名稱的 MD5 雜湊字串，藉此為 CREATE USER 或 ALTER USER 命令指定密碼。	2016 年 4 月 21 日
STV_TBL_PERM 中的新資料欄	backup 系統檢視中的 STV_TBL_PERM 資料欄指出叢集快照中是否包含資料表。如需詳細資訊，請參閱BACKUP。	2016 年 4 月 21 日
不備份資料表	對於不包含重要資料的資料表 (例如臨時資料表)，您可以在 CREATE TABLE 或 CREATE TABLE AS 陳述式中指定 BACKUP NO，以防止 Amazon Redshift 將該資料表納入自動或手動快照中。使用不備份資料表可以在建立快照以及從快還原照時節省處理時間，並節省 Amazon S3 上的儲存空間。	2016 年 4 月 7 日
VACUUM 刪除閾值	根據預設，VACUUM 命令現在會回收空間，使至少百分之 95 的剩餘資料列不會被標記為刪除。因此，與 100% 回收刪除的資料列相比，VACUUM 通常在刪除階段需要的時間更少。您可以在執行 VACUUM 命令時加入 TO threshold PERCENT 參數，來變更單一資料表的預設閾值。	2016 年 4 月 7 日
SVV_TRANSACTIONS 系統資料表	SVV_TRANSACTIONS 系統檢視會記錄資料庫中目前保持鎖定之資料表的交易相關資訊。	2016 年 4 月 7 日

變更	描述	變更日期
使用 IAM 角色存取其他 AWS 資源	若要在叢集和另一個 AWS 資源 (例如 Amazon S3、DynamoDB、Amazon EMR 或 Amazon EC2) 之間移動資料，叢集必須具備存取此資源和執行必要動作的許可。您現在可以指定叢集用於身分驗證和授權的 IAM 角色，這是使用 COPY、UNLOAD 或 CREATE LIBRARY 命令提供存取金鑰的更安全替代方法。如需詳細資訊，請參閱角色類型存取控制。	2016 年 3 月 29 日
VACUUM 排序閾值	現在，若資料表中超過百分之 95 的資料列已排序，VACUUM 命令會略過資料表的排序階段。您可以在執行 VACUUM 命令時加入 TO threshold PERCENT 參數，來變更單一資料表的預設排序閾值。	2016 年 3 月 17 日
STL_CONNECTION_LOG 中的新資料欄	STL_CONNECTION_LOG 系統資料表有三個新資料欄。如果您定期將稽核日誌載入 Amazon Redshift 資料表，則需要將以下新欄位新增到目標資料表：sslversion、sslcipher 和 mtu。	2016 年 3 月 17 日
使用 bzip2 壓縮進行 UNLOAD	您現在可以選擇使用 bzip2 壓縮進行 UNLOAD。	2016 年 2 月 8 日
ALTER TABLE APPEND	ALTER TABLE APPEND 會從現有來源資料表移出資料，將資料列附加到目標資料表。因為是移動而不是複製資料，ALTER TABLE APPEND 通常比類似的 CREATE TABLE AS 或 INSERT INTO 操作更快。	2016 年 2 月 8 日
WLM 查詢佇列跳轉	如果工作負載管理 (WLM) 因為逾時而取消唯讀查詢 (如 SELECT 陳述式)，WLM 會嘗試將查詢路由到下一個相符的佇列。如需更多資訊，請參閱 WLM 查詢佇列跳轉	2016 年 1 月 7 日
ALTER DEFAULT PRIVILEGES	您可以使用 ALTER DEFAULT PRIVILEGES 命令來定義日後由指定使用者建立的物件將套用的存取權限預設設定。	2015 年 12 月 10 日
bzip2 檔案壓縮	COPY 命令支援從使用 bzip2 壓縮的檔案載入資料。	2015 年 12 月 10 日

變更	描述	變更日期
NULLS FIRST 和 NULLS LAST	您可以指定 ORDER BY 子句要將 NULLS 排在結果集中的開頭或結尾。如需詳細資訊，請參閱 ORDER BY 子句 及 範圍函數語法摘要。	2015 年 11 月 19 日
CREATE LIBRARY 的 REGION 關鍵字	如果包含 UDF 程式庫檔案的 Amazon S3 儲存貯體不是位於 Amazon Redshift 叢集所在的同一 AWS 區域，您可以使用 REGION 選項來指定資料所在的區域。如需詳細資訊，請參閱CREATE LIBRARY。	2015 年 11 月 19 日
使用者定義的純量函數 (UDF)	您現在可以建立自訂的使用者定義純量函數來實作非 SQL 處理的功能，這些功能可以是由 Python 2.7 標準程式庫中支援 Amazon Redshift 的模組提供，或是由您自己以 Python 程式語言編寫的自訂 UDF 提供。如需詳細資訊，請參閱建立使用者定義的函數。	2015 年 9 月 11 日
WLM 組態中的動態屬性	WLM 組態參數現在支援動態套用部分屬性。其他屬性仍然是靜態變更，需要將關聯的叢集重新啟動才會套用組態變更。如需詳細資訊，請參閱 WLM 動態和靜態組態屬性 及 實作工作負載管理。	2015 年 8 月 3 日
LISTAGG 函數	LISTAGG 函數 和 LISTAGG 範圍函數 會傳回字串，這個字串是將一組資料欄的值串連起來建立而成。	2015 年 7 月 30 日
已棄用的參數	已棄用 max_cursor_result_set_size 組態參數。游標結果集的大小限制是根據叢集的節點類型而定。如需詳細資訊，請參閱游標限制條件。	2015 年 7 月 24 日
已已修 COPY 命令的文件	大幅修訂 COPY 命令的參考資料，使資料易懂更容易取得。	2015 年 7 月 15 日
從 Avro 格式 COPY	COPY 命令支援從 Amazon S3、Amazon EMR 上的資料檔案和從遠端主機 (透過 SSH 連線) 載入 Avro 格式的資料。如需詳細資訊，請參閱 AVRO 及 從 Avro 複製的範例。	2015 年 7 月 8 日

變更	描述	變更日期
STV_START UP_RECOVE RY_STATE	STV_STARTUP_RECOVERY_STATE 系統資料表會記錄在叢集重新啟動操作期間暫時鎖定的資料表狀態。Amazon Redshift 會在處理資料表時暫時鎖定資料表，以解決叢集重新啟動後的過時交易。	2015 年 5 月 25 日
排名函數可選用 ORDER BY 選項	現在，某些排名視窗函數的 ORDER BY 選項為選用。如需詳細資訊，請參閱 CUME_DIST 範圍函數、DENSE_RANK 範圍函數、RANK 範圍函數、NTILE 範圍函數、PERCENT_RANK 範圍函數、ROW_NUMBER 範圍函數。	2015 年 5 月 25 日
交錯的排序索引鍵	交錯的排序索引鍵使排序索引鍵中的每個資料欄具有相等權重。若使用交錯的排序索引鍵來代替預設的複合索引鍵，當查詢對第二個排序資料欄使用限制性述詞，尤其是查詢大型資料表時，可以大幅改善查詢效能。在對相同資料表中不同資料欄套用多個篩選條件時，交錯排序也能改善效能。如需詳細資訊，請參閱 使用排序索引鍵 及 CREATE TABLE。	2015 年 5 月 11 日
已修訂的調校查詢效能主題	已擴充調校查詢效能的內容，增加用於分析查詢效能的新查詢以及更多範例。此外，也將主題修訂得更清楚更完整。設計查詢的 Amazon Redshift 最佳實務提供如何撰寫查詢以改善效能的詳細資訊。	2015 年 3 月 23 日
SVL_QUERY _QUEUE_INFO	SVL_QUERY_QUEUE_INFO 檢視摘要說明在 WLM 查詢佇列或遞交佇列中花費時間之查詢的詳細資訊。	2015 年 2 月 19 日
SVV_TABLE _INFO	您可以使用 SVV_TABLE_INFO 檢視來診斷和解決可能影響查詢效能的資料表設計問題，包括壓縮編碼、分佈索引鍵、排序樣式、資料配送偏度、資料表大小、統計等方面的問題。	2015 年 2 月 19 日
UNLOAD 使用伺服器端檔案加密	UNLOAD 命令現在會自動使用 Amazon S3 伺服器端加密 (SSE) 來加密所有卸載資料檔案。伺服器端加密增加了另一層資料安全，且對效能幾乎沒有影響。	2014 年 10 月 31 日

變更	描述	變更日期
CUME_DIST 範圍函數	CUME_DIST 範圍函數 會計算視窗或分割區內值的累積分佈。	2014 年 10 月 31 日
MONTHS_BETWEEN 函數	MONTHS_BETWEEN 函數 可判斷兩個日期之間有幾個月。	2014 年 10 月 31 日
NEXT_DAY 函數	NEXT_DAY 函數 會傳回在給定日期之後的指定那天 (星期幾) 後的第一個執行個體的日期。	2014 年 10 月 31 日
PERCENT_RANK 範圍函數	PERCENT_RANK 範圍函數 會計算給定資料列的百分比排行。	2014 年 10 月 31 日
RATIO_TO_REPORT 範圍函數	RATIO_TO_REPORT 範圍函數 會計算視窗或分割區內值與值總和的比率。	2014 年 10 月 31 日
TRANSLATE 函數	TRANSLATE 函數 會用指定的取代值來取代給定表達式中出現的所有指定字元。	2014 年 10 月 31 日
NVL2 函數	NVL2 函數 會根據指定的表達式判斷值為 NULL 還是 NOT NULL，傳回兩個值中的一個。	2014 年 10 月 16 日
MEDIAN 範圍函數	MEDIAN 範圍函數 會計算視窗或分割區內值範圍的中位數。	2014 年 10 月 16 日
GRANT 和 REVOKE 命令的 ON ALL TABLES IN SCHEMA schema_name 子句	GRANT 和 REVOKE 命令已更新為帶有 ON ALL TABLES IN SCHEMA schema_name 子句。這個子句可讓您使用單一命令來變更結構描述中所有資料表的權限。	2014 年 10 月 16 日

變更	描述	變更日期
DROP SCHEMA、DROP TABLE、DROP USER 和 DROP VIEW 命令的 IF EXISTS 子句	DROP SCHEMA、DROP TABLE、DROP USER、DROP VIEW 命令已更新為帶有 IF EXISTS 子句。如果指定的物件不存在，這個子句可使命令不進行任何變更並傳回訊息，而不是以發生錯誤終止。	2014 年 10 月 16 日
CREATE SCHEMA 和 CREATE TABLE 命令的 IF NOT EXISTS 子句	CREATE SCHEMA 和 CREATE TABLE 命令已更新為帶有 IF NOT EXISTS 子句。如果指定的物件已存在，這個子句可使命令不進行任何變更並傳回訊息，而不是以發生錯誤終止。	2014 年 10 月 16 日
COPY 支援 UTF-16 編碼	COPY 現在支援從使用 UTF-16 編碼和 UTF-8 編碼的資料檔案載入資料。如需詳細資訊，請參閱ENCODING。	2014 年 9 月 29 日
新的工作負載管理教學	教學：設定手動工作負載管理 (WLM) 佇列引導您完成設定工作負載管理 (WLM) 佇列的過程，以改進查詢處理和資源配置。	2014 年 9 月 25 日
AES 128 位元加密	現在，COPY 命令從透過 Amazon S3 用戶端加密所加密的資料檔案載入時，支援 AES 128 位元加密和 AES 256 位元加密。如需詳細資訊，請參閱從 Amazon S3 載入加密的資料檔案。	2014 年 9 月 29 日
PG_LAST_UNLOAD_COUNT 函數	PG_LAST_UNLOAD_COUNT 函數會傳回最近一次 UNLOAD 操作處理的資料列數。如需詳細資訊，請參閱PG_LAST_UNLOAD_COUNT。	2014 年 9 月 15 日
新的故障診斷查詢章節	對查詢進行故障診斷 提供快速的參考，方便識別和解決您使用 Amazon Redshift 查詢時可能遇到的一些最常見和最嚴重的問題。	2014 年 7 月 7 日

變更	描述	變更日期
新的載入資料教學	教學課程：從 Amazon S3 載入資料 將從頭到尾引導您完成從 Amazon S3 儲存貯體中的資料檔案將資料載入 Amazon Redshift 資料庫資料表的過程。	2014 年 7 月 1 日
PERCENTILE_CONT 範圍函數	PERCENTILE_CONT 範圍函數 是反向分發函數，假定連續的分發模型。它採用百分位數值和排序規格，且會傳回插入值，該值將根據排序規格落入給定的百分位數值。	2014 年 6 月 30 日
PERCENTILE_DISC 範圍函數	PERCENTILE_DISC 範圍函數 是反向分發函數，假定不連續的分發模型。它採用百分位數值和排序規格，且會傳回集裡的一個元素。	2014 年 6 月 30 日
GREATEST 和 LEAST 函數	GREATEST 和 LEAST 函數 函數會傳回表達式清單中的最大或最小值。	2014 年 6 月 30 日
跨區域 COPY	COPY 命令支援從 Amazon Redshift 叢集以外區域中的 Amazon S3 儲存貯體或 Amazon DynamoDB 資料表載入資料。如需詳細資訊，請參閱 COPY 命令參考資料中的REGION。	2014 年 6 月 30 日
擴充最佳實務	Amazon Redshift 最佳實務的內容已擴充、重新組織並移至導覽階層的頂端，使其更容易被找到。	2014 年 5 月 28 日
UNLOAD 至單一檔案	UNLOAD 命令可以選擇性加入 PARALLEL OFF 選項，將資料表資料依序卸載到 Amazon S3 上的單一檔案。如果資料大於檔案大小上限 6.2 GB，UNLOAD 會額外建立檔案。	2014 年 5 月 6 日
REGEXP 函數	REGEXP_COUNT、REGEXP_INSTR、REGEXP_REPLACE 函數會根據正規表達式模式比對來操作字串。	2014 年 5 月 6 日
從 Amazon EMR 進行 COPY	COPY 命令支援直接從 Amazon EMR 叢集載入資料。如需詳細資訊，請參閱從 Amazon EMR 載入資料。	2014 年 4 月 18 日

變更	描述	變更日期
WLM 並行限制提高	現在您可以設定工作負載管理 (WLM) 在使用者定義的佇列中同時執行最多 50 個查詢。限制提高後，使用者可以修改 WLM 組態，更靈活地管理系統效能。如需更多資訊，請參閱 實作手動 WLM	2014 年 4 月 18 日
新的組態參數用於管理游標大小	`max_cursor_result_set_size` 組態參數定義了針對大型查詢的每個游標結果集可以傳回的數據大小上限 (以 MB 為單位)。此參數值也會影響叢集的並行游標數，可讓您配置一個值來增加或減少叢集的游標數。 如需詳細資訊，請參閱本指南中的 DECLARE 和《Amazon Redshift 管理指南》中的設定游標結果集的大小上限。	2014 年 3 月 28 日
從 JSON 格式 COPY	COPY 命令支援從 Amazon S3 上的資料檔案和從遠端主機 (透過 SSH 連線) 載入 JSON 格式的資料。如需詳細資訊，請參閱從 JSON 格式 COPY 使用須知。	2014 年 3 月 25 日
新的系統資料表 STL_PLAN_INFO	STL_PLAN_INFO 資料表補充了 EXPLAIN 命令，是查看查詢計劃的另一種方式。	2014 年 3 月 25 日
新函數 REGEXP_SUBSTR	REGEXP_SUBSTR 函數 會搜索正規表達式模式，傳回從字串中擷取的字元。	2014 年 3 月 25 日
STL_COMMIT_STATS 的新資料欄	STL_COMMIT_STATS 資料表有兩個新資料欄：`numxids` 和 `oldestxid` 。	2014 年 3 月 6 日
從 SSH COPY 支援 gzip 和 lzop	透過 SSH 連線載入資料時，COPY 命令支援 gzip 和 lzop 壓縮。	2014 年 2 月 13 日

變更	描述	變更日期
新函數	ROW_NUMBER 範圍函數 會傳回目前資料列的編號。STRTOL 函數 將指定底數之數字的字串表達式轉換為等同的整數值。PG_CANCEL_BACKEND 和 PG_TERMINATE_BACKEND 可讓使用者取消查詢和工作階段連線。新增 LAST_DAY 函數以便和 Oracle 相容。	2014 年 2 月 13 日
新的系統資料表	STL_COMMIT_STATS 系統資料表提供與遞交效能相關的指標，包括各種遞交階段的時間，以及遞交的區塊數。	2014 年 2 月 13 日
FETCH 單一節點的叢集	在單一節點的叢集上使用游標時，使用 FETCH 命令可以擷取的資料列數上限為 1000。單一節點的叢集不支援 FETCH FORWARD ALL。	2014 年 2 月 13 日
DS_DIST_ALL_INNER 重新配送策略	Explain 計劃輸出中的 DS_DIST_ALL_INNER 指出整個內部資料表已重新配送至單一配量，因為外部資料表使用 DISTSTYLE ALL。如需詳細資訊，請參閱 聯結類型範例 及 評估查詢計畫。	2014 年 1 月 13 日
用於查詢的新系統資料表	Amazon Redshift 新增新的系統資料表，客戶可用來評估查詢執行，以便進行調校和故障診斷。如需詳細資訊，請參閱 SVL_COMPILE, STL_SCAN, STL_RETURN, STL_SAVE STL_ALERT_EVENT_LOG.	2014 年 1 月 13 日
單一節點的游標	單一節點的叢集現在支援游標。單一節點的叢集可同時有兩個游標，其結果集大小上限為 32 GB。建議您在單一節點的叢集上，將 ODBC 快取大小參數設為 1,000。如需詳細資訊，請參閱DECLARE。	2013 年 13 月 12 日
ALL 配送樣式	ALL 配送可以動態縮短某些查詢類型的執行時間。當資料表使用 ALL 配送樣式時，會將資料表的副本配送至每個節點。由於資料表與其他每個資料表有效地共置，因此在查詢執行期間不需要重新配送。ALL 配送並不適合所有資料表，因為會增加儲存空間的需求，以及增加載入時間。如需詳細資訊，請參閱使用資料分佈樣式。	2013 年 11 月 11 日

變更	描述	變更日期
從遠端主機 COPY	除了從 Amazon S3 和 Amazon DynamoDB 資料表上的資料檔案載入資料表之外，COPY 命令還可以使用 SSH 連線，從 Amazon EMR 叢集、Amazon EC2 執行個體和其他遠端主機載入文字資料。Amazon Redshift 會使用多個同時的 SSH 連線，以平行方式讀取和載入資料。如需詳細資訊，請參閱從遠端主機載入資料。	2013 年 11 月 11 日
使用的 WLM 記憶體百分比	您可以為工作負載管理 (WLM) 組態中每個佇列的記憶體指定特定百分比，來平衡工作負載。如需詳細資訊，請參閱實作手動 WLM。	2013 年 11 月 11 日
APPROXIMATE COUNT(DISTINCT)	使用 APPROXIMATE COUNT(DISTINCT) 的查詢執行較快，其相對錯誤率約 2%。近似計數（不同）函數使用 HyperLogLog 算法。如需更多資訊，請參閱 COUNT 函數。	2013 年 11 月 11 日
可擷取最近查詢詳細資訊的新 SQL 函數	四個新的 SQL 函數可擷取最近的查詢和 COPY 命令的詳細資訊。新函數使查詢系統日誌資料表變得更加容易，且在許多情況下不必存取系統資料表即可提供必要的詳細資訊。如需詳細資訊，請參閱 PG_BACKEND_PID, PG_LAST_COPY_ID, PG_LAST_COPY_COUNT, PG_LAST_QUERY_ID.	2013 年 11 月 1 日
UNLOAD 的 MANIFEST 選項	UNLOAD 命令的 MANIFEST 選項補強了 COPY 命令的 MANIFEST 選項。使用 UNLOAD 搭配 MANIFEST 選項會自動建立資訊清單檔案，檔案中明確列出在 Amazon S3 上由卸載操作 (UNLOAD) 建立的資料檔案。接著您可以對相同資訊清單檔案使用 COPY 命令來載入資料。如需詳細資訊，請參閱 將資料卸載到 Amazon S3 及 UNLOAD 範例。	2013 年 11 月 1 日
COPY 的 MANIFEST 選項	您可以在 COPY 命令中使用 MANIFEST 選項，明確列出將從 Amazon S3. 載入的資料檔案。	2013 年 10 月 18 日

變更	描述	變更日期
用於查詢的故障診斷的系統資料表	新增用於對查詢進行故障診斷的系統資料表的文件。用於記錄日誌的 STL 檢視小節現在包含下列系統資料表的文件：STL_AGGR、STL_BCAST、STL_DIST、STL_DELETE、STL_HASH、STL_HASHJOIN、STL_INSERT、STL_LIMIT、STL_MERGE、STL_MERGEJOIN、STL_NESTLOOP、STL_PARSE、STL_PROJECT、STL_SCAN、STL_SORT、STL_UNIQUE、STL_WINDOW。	2013 年 10 月 3 日
CONVERT_TIMEZONE 函數	CONVERT_TIMEZONE 函數 會將時間戳記從一個時區轉換為另一個時區，並可選擇自動調整日光節約時間。	2013 年 10 月 3 日
SPLIT_PART 函數	SPLIT_PART 函數 會在指定分隔符號之處分割字串，並傳回指定位置的部分。	2013 年 10 月 3 日
STL_USERLOG 系統資料表	當建立、更改或刪除資料庫使用者時，STL_USERLOG 會記錄發生變更的詳細資訊。	2013 年 10 月 3 日
LZO 資料欄編碼和 LZOP 檔案壓縮。	LZO 資料欄壓縮編碼具有非常高的壓縮率和良好的效能。從 Amazon S3 COPY 支援從使用 LZOP 壓縮的檔案載入資料。	2013 年 9 月 19 日
JSON、常規表達式及游標	新增剖析 JSON 字串的支援，包括使用常規表達式進行模式比對、以及透過 ODBC 連線使用游標擷取大型資料集。如需詳細資訊，請參閱 JSON 函數、模式比對條件 及 DECLARE。	2013 年 9 月 10 日
COPY 的 ACCEPTINVCHAR 選項	使用 COPY 命令並指定 ACCEPTINVCHAR 選項，可以成功載入包含無效 UTF-8 字元的資料。	2013 年 8 月 29 日
COPY 的 CSV 選項	COPY 命令現在支援從 CSV 格式的輸入檔案載入資料。	2013 年 8 月 9 日
CRC32	CRC32 函數 會執行循環冗餘檢查。	2013 年 8 月 9 日

變更	描述	變更日期
WLM 萬用字元	在將使用者群組和查詢群組新增至佇列時，工作負載管理 (WLM) 支援使用萬用字元。如需詳細資訊，請參閱萬用字元。	2013 年 8 月 1 日
WLM 逾時	若要限制給定的 WLM 佇列中允許查詢使用的時間，您可以設定每個佇列的 WLM 逾時值。如需詳細資訊，請參閱WLM 逾時。	2013 年 8 月 1 日
新的 COPY 選項 'auto' 和 'epochsecs'	COPY 命令會執行日期和時間格式的自動辨識。新的時間格式 'epochsecs' 和 'epochmillisecs' 讓 COPY 可以載入 epoch 格式的資料。	2013 年 7 月 25 日
CONVERT_TIMEZONE 函數	CONVERT_TIMEZONE 函數 可將時間戳記從一個時區轉換為另一個時區。	2013 年 7 月 25 日
FUNC_SHA1 函數	FUNC_SHA1 函數 使用 SHA1 演算法轉換字串。	2013 年 7 月 15 日
max_execution_time	若要限制給定的 WLM 佇列中允許查詢使用的時間量，您可以在 WLM 組態中設定 max_execution_time 參數。如需詳細資訊，請參閱修改 WLM 組態。	2013 年 7 月 22 日
四個位元組的 UTF-8 字元	VARCHAR 資料類型現在支援四個位元組的 UTF-8 字元。不支援五個位元組或更長的 UTF-8 字元。如需詳細資訊，請參閱儲存與範圍。	2013 年 7 月 18 日
SVL_QERROR	已棄用 SVL_QERROR 系統檢視。	2013 年 7 月 12 日
修訂文件歷史記錄	Document History (文件歷史記錄) 頁面現在會顯示文件更新的日期。	2013 年 7 月 12 日
STL_UNLOAD_LOG	STL_UNLOAD_LOG 會記錄卸載操作的詳細資訊。	2013 年 7 月 5 日
JDBC 擷取大小參數	若要在使用 JDBC 擷取大型資料時避免用戶端發生記憶體不足的錯誤，可以設定 JDBC 擷取大小參數來啟用用戶端的批次資料擷取。如需詳細資訊，請參閱設定 JDBC 擷取大小參數。	2013 年 6 月 27 日

變更	描述	變更日期
UNLOAD 加密的檔案	UNLOAD 現在支援將資料表的資料卸載到 Amazon S3 上已加密的檔案。	2013 年 5 月 22 日
臨時憑證	COPY 和 UNLOAD 現在支援使用暫時性登入資料。	2013 年 11 月 4 日
補充說明	闡明並已擴充設計資料表和載入資料的討論。	2013 年 2 月 14 日
新增最佳實務	新增設計資料表的 Amazon Redshift 最佳實務和載入資料的 Amazon Redshift 最佳實務。	2013 年 2 月 14 日
闡明密碼限制	闡明了 CREATE USER 和 ALTER USER 的密碼限制，各種小修訂。	2013 年 2 月 14 日
新指南	這是《Amazon Redshift 開發人員指南》的第一個版本。	2013 年 2 月 14 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。