aws

查詢 Amazon Redshift 的最佳實務

AWS 方案指引

AWS 方案指引: 查詢 Amazon Redshift 的最佳實務

Table of Contents

查詢 Amazon Redshift 的最佳實務

Ethan Stark，Amazon Web Services (AWS)

2024 年 6 月 (文件歷史記錄)

概觀

本指南提供在 Amazon Redshift 中最佳化查詢和資料表效能的建議和最佳實務。您可以使用 Amazon Redshift 來查詢資料倉儲和資料湖中 PB 的結構化和半結構化資料。本指南也提供 Amazon Redshift 資料倉儲核心架構元件的概觀。這些知識以及對資料表屬性、叢集組態和查詢結構等查詢效能因素的了解，可協助您為 Amazon Redshift 資料倉儲設計高效且有效的資料表和查詢。

目標對象

本指南適用於在 Amazon Redshift 中設計或使用資料表和查詢的資料工程師、資料架構師和資料分析師。

目標

本指南可協助您和組織達成下列目標：

- 設計最佳資料儲存和擷取操作的資料表
- 設計查詢以獲得最佳效能和節省成本
- 最佳化 Amazon Redshift Spectrum 的效能，以直接從 Amazon Simple Storage Service (Amazon S3) 上的檔案查詢資料

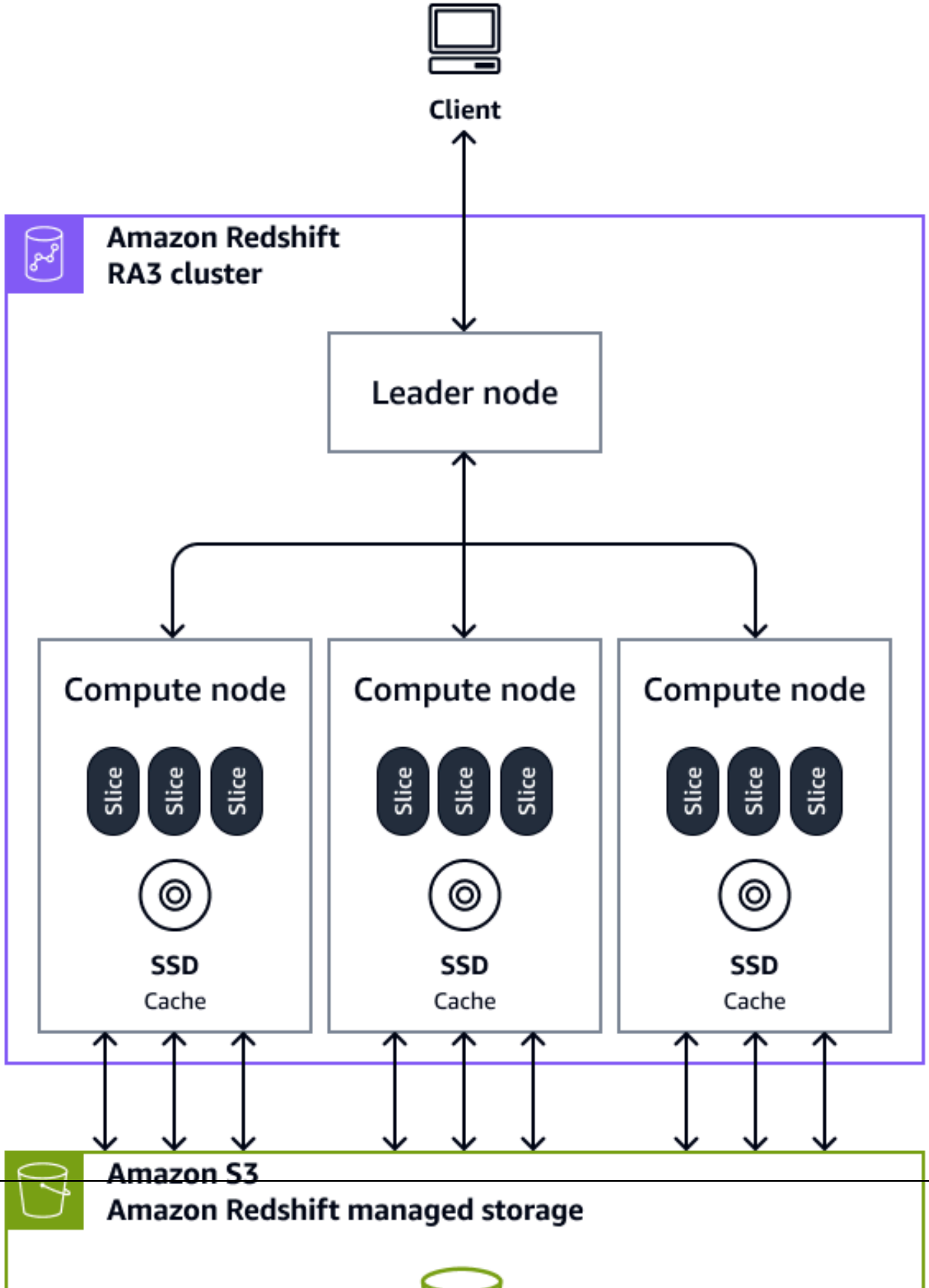
Amazon Redshift 資料倉儲的架構元件

我們建議您對 Amazon Redshift 資料倉儲中的核心架構元件有基本的了解。這些知識可協助您進一步了解如何設計查詢和資料表，以獲得最佳效能。

Amazon Redshift 中的資料倉儲包含下列核心架構元件：

- 叢集 – 叢集由一或多個運算節點組成，是 Amazon Redshift 資料倉儲的核心基礎設施元件。運算節點對外部應用程式是透明的，但您的用戶端應用程式只會直接與領導節點互動。典型叢集有兩個或多個運算節點。運算節點會透過領導節點進行協調。
- 領導節點 – 領導節點管理用戶端程式和所有運算節點的通訊。領導者節點也會準備每當查詢提交至叢集時執行查詢的計劃。當計劃準備就緒時，領導節點會編譯程式碼、將編譯的程式碼分發給運算節點，然後將資料配量指派給每個運算節點來處理查詢結果。
- 運算節點 – 運算節點會執行查詢。領導節點會針對計畫的個別元素編譯程式碼，以執行查詢並將程式碼指派給個別運算節點。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。每個運算節點都有自己的專用 CPU、記憶體和連接的磁碟儲存。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和儲存容量。
- 節點配量 – 運算節點會分割成稱為配量的單位。運算節點中的每個配量都會配置一部分的節點記憶體和磁碟空間，用於處理指派給節點的部分工作負載。這些分割接著會平行運作，以完成操作。資料會根據特定資料表的分佈樣式和分佈索引鍵，在配量之間分佈。資料的均勻分佈可讓 Amazon Redshift 將工作負載平均指派給配量，並最大化平行處理的優勢。每個運算節點的配量數目取決於節點的類型。如需詳細資訊，請參閱 Amazon Redshift 文件中的 Amazon Redshift 中的叢集和節點 。
- 大規模平行處理 (MPP) – Amazon Redshift 使用 MPP 架構快速處理資料，甚至是複雜的查詢和大量的資料。多個運算節點會在部分資料上執行相同的查詢程式碼，以最大化平行處理。
- 用戶端應用程式 – Amazon Redshift 與各種資料載入、擷取、轉換和載入 (ETL)、商業智慧 (BI) 報告、資料探勘和分析工具整合。所有用戶端應用程式只會透過領導節點與叢集通訊。

下圖顯示 Amazon Redshift 資料倉儲的架構元件如何協同運作以加速查詢。


Client
Amazon Redshift
RA3 cluster
Leader node
Compute node
Slice
Slice
Slice
SSD
Cache
Compute node
Slice
Slice
Slice
SSD
Cache
Compute node
Slice
Slice
Slice
SSD
Cache
Amazon S3
Amazon Redshift managed storage

查詢生命週期有七個階段：

1. 查詢接收和剖析：
   - 領導者節點接收查詢並剖析 SQL。
   - 剖析器會產生初始查詢樹狀結構，代表原始查詢的邏輯結構。
   - Amazon Redshift 將此查詢樹饋送至查詢最佳化工具。
2. 查詢最佳化：
   - 最佳化工具會評估查詢，並視需要將其重寫以最大化效率。
   - 此最佳化程序可能涉及建立多個相關查詢來取代單一查詢。
3. 查詢計畫產生：
   - 最佳化工具會產生查詢計畫（或多個計畫，如有需要）以供執行。
   - 查詢計畫會指定執行選項，例如聯結類型、聯結順序、彙總方法和資料分佈需求。
4. 執行引擎翻譯：
   - 執行引擎會將查詢計畫轉換為分散的步驟、區段和串流：
     - 步驟 – 代表查詢執行期間所需的個別操作。可以合併步驟，以允許運算節點執行查詢、聯結或其他資料庫操作。
     - 客群 – 結合單一程序可執行的數個步驟。這是運算節點配量可執行的最小編譯單位。（配量是 Amazon Redshift 中平行處理的單位。）
     - 串流 – 分散在可用運算節點配量的區段集合。
   - 執行引擎會根據這些步驟、區段和串流產生編譯的程式碼。編譯的程式碼執行速度比解譯的程式碼更快，並消耗較少的運算容量。
   - 領導節點會將編譯的程式碼廣播至運算節點。
5. 平行執行：
   - 每個串流都會執行此步驟一次。
   - 運算節點配量會平行執行查詢區段。
   - 在此過程中，Amazon Redshift 會最佳化網路通訊、記憶體用量和磁碟管理，將中繼結果從一個查詢計畫步驟傳遞到下一個步驟。
   - 此最佳化有助於加快查詢執行速度。
6. 串流處理：
   - 每個串流都會執行此步驟一次。
   - 引擎會為每個串流建立可執行區段，以有效率地平行處理。
7. 最終排序和彙總：

- 領導節點會處理查詢所需的任何最終排序或彙總。
- 完成後，領導節點會將結果傳回給用戶端。

如需架構元件的資訊，請參閱 Amazon Redshift 文件中的資料倉儲系統架構。

Amazon Redshift 的查詢效能因素

有一些因素會影響查詢效能。資料、叢集和資料庫操作的下列層面都有助於查詢的處理速度：

- 資料表屬性
  - 排序索引鍵 (Amazon Redshift Advisor)
  - 資料壓縮（自動化）
  - 資料分佈（自動化）
  - 資料表維護（自動化）
- 叢集組態
  - Node type
  - 節點大小、節點數量和配量
  - 工作負載管理（自動化）
  - 短期查詢加速（自動化）
- SQL 查詢
  - 查詢結構
  - 程式碼編譯

資料表屬性

Amazon Redshift 資料表是將資料存放在 Amazon Redshift 中的基本單位，而且每個資料表都有一組屬性來決定其行為和可存取性。這些屬性包括排序、分佈樣式、壓縮編碼等。了解這些屬性對於最佳化 Amazon Redshift 資料表的效能、安全性和成本效益至關重要。

排序索引鍵

Amazon Redshift 會根據資料表的排序索引鍵，以排序順序將資料存放在磁碟上。查詢最佳化工具與查詢處理器會使用資料位於運算節點中位置的相關資訊，以減少必須掃描的區塊數量。這可透過減少要處理的資料量，大幅改善查詢速度。我們建議您使用排序索引鍵來協助 `WHERE`子句中的篩選條件。如需詳細資訊，請參閱《Amazon Redshift 文件》中的使用排序索引鍵。

資料壓縮

資料壓縮可減少儲存需求，進而減少磁碟 I/O 並改善查詢效能。當您執行查詢時，壓縮的資料會讀取至記憶體，然後在查詢執行時解壓縮。透過將較少的資料載入記憶體，Amazon Redshift 可以配置更多

記憶體來分析資料。由於單欄式儲存會循序存放類似的資料，因此 Amazon Redshift 可以套用專門與單欄式資料類型綁定的調整式壓縮編碼。在資料表資料欄上啟用資料壓縮的最佳方法是使用 Amazon Redshift 中的 AUTO選項，在您使用資料載入資料表時套用最佳壓縮編碼。若要進一步了解如何使用自動資料壓縮，請參閱《Amazon Redshift 文件》中的使用自動壓縮載入資料表。

資料分佈

Amazon Redshift 會根據資料表的分佈樣式將資料存放在運算節點上。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。我們建議您使用分佈索引鍵來促進最常見的聯結。如需詳細資訊，請參閱《Amazon Redshift 文件》中的使用資料分佈樣式。

資料表維護

雖然 Amazon Redshift 為大多數工作負載提供業界領先的立即可用效能，但保持 Amazon Redshift 叢集正常運作需要維護。更新和刪除資料會建立必須清空的無效資料列，而且如果附加順序與排序索引鍵不一致，甚至必須重新排序僅附加資料表。

Vacuum

Amazon Redshift 中的清空程序對於 Amazon Redshift 叢集的運作狀態和維護至關重要。它也會影響查詢的效能。由於 刪除和更新會同時標記舊資料，但未實際移除，因此您必須使用清空來回收先前 UPDATE和 DELETE操作標記為刪除的資料表列佔用的磁碟空間。Amazon Redshift 可以在背景的資料表上自動排序和執行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。如果資料表具有排序索引鍵，且資料表負載未最佳化，無法在插入時排序，則您必須使用清空來排序資料 （這對於效能至關重要）。如需詳細資訊，請參閱 Amazon Redshift 文件中的清空資料表。

分析

ANALYZE 操作會更新 Amazon Redshift 資料庫中資料表的統計中繼資料。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 會持續監控資料庫，並自動在背景中執行分析操作。為了將對系統效能的影響降至最低，ANALYZE操作會在工作負載較輕的期間自動執行。如果您選擇明確執行 ANALYZE，請執行下列動作：

- 在執行查詢之前執行 ANALYZE命令。
- 在每個定期載入或更新週期結束時，定期在資料庫上執行 ANALYZE命令。

- 在您建立的新資料表和經歷重大變更的現有資料表或資料欄上執行 ANALYZE命令。
- 考慮在不同類型的資料表和資料欄的不同排程上執行ANALYZE操作，取決於它們在查詢中的使用及其變更的傾向。
- 若要節省時間和叢集資源，請在執行 ANALYZE命令時使用 PREDICATE COLUMNS子句。

叢集組態

叢集是執行實際儲存和處理資料的節點集合。如果您想要達成下列目標，以正確的方式設定 Amazon Redshift 叢集至關重要：

- 高可擴展性和並行性
- 有效使用 Amazon Redshift
- 更好的效能
- 降低成本

Node type

Amazon Redshift 叢集可以使用多種節點類型 (RA3、DC2 和 DS2) 的其中之一。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。成本和效能最佳化從選擇正確的節點類型和大小開始。如需節點類型的詳細資訊，請參閱 Amazon Redshift 文件中的 Amazon Redshift 叢集概觀。

節點大小、節點數量和配量

運算節點可分割為分割。更多節點表示更多的處理器和配量，這可讓您的查詢透過跨配量同時執行部分查詢來更快地處理。不過，更多節點也表示費用更高。這表示您必須找到適合您系統的成本和效能平衡。如需 Amazon Redshift 叢集架構的詳細資訊，請參閱 Amazon Redshift 文件中的資料倉儲系統架構。

工作負載管理

Amazon Redshift 工作負載管理 (WLM) 可讓使用者彈性地管理具有優先順序的工作負載佇列，因此短期、快速執行的查詢不會卡在長時間執行的查詢後方的佇列中。自動 WLM 使用機器學習 (ML) 演算法來描述查詢，並將其放置在具有適當資源的適當佇列中，同時管理查詢並行和記憶體配置。如需 WLM 的詳細資訊，請參閱 Amazon Redshift 文件中的實作工作負載管理。

短期查詢加速

短期查詢加速 (SQA) 會優先考慮短期執行的查詢，而不是長時間執行的查詢。SQA 會在專用空間中執行查詢，因此 SQA 查詢不會強制在較長查詢後方的佇列中等待。SQA 只優先處理短期執行且位於使用者定義佇列中的查詢。如果您使用 SQA，短期執行的查詢會更快開始執行，而且您可以更快看到結果。如果您啟用 SQA，您可以減少或消除專用於短期執行查詢的 WLM 佇列。此外，長時間執行的查詢不需要爭用 WLM 佇列中的槽。這表示您可以將 WLM 佇列設定為使用較少的查詢槽。如果您使用較低的並行，查詢輸送量會提高，且大多數工作負載的整體系統效能也會獲得改善。如需 SQA 的詳細資訊，請參閱《Amazon Redshift 文件》中的使用短期查詢加速。

SQL 查詢

資料庫查詢是從資料庫取得資料的請求。請求應使用 SQL 出現在 Amazon Redshift 叢集中。Amazon Redshift 支援透過 Java Database Connectivity (JDBC) 和 Open Database Connectivity (ODBC) 連線的 SQL 用戶端工具。您可以使用支援 JDBC 或 ODBC 驅動程式的大多數 SQL 用戶端工具。

查詢結構

您的查詢寫入方式會大幅影響其效能。我們建議您撰寫查詢，以視需要處理和傳回最少的資料，以符合您的需求。如需如何建構查詢的詳細資訊，請參閱本指南的設計 Amazon Redshift 查詢最佳實務一節。

程式碼編譯

Amazon Redshift 會為每個查詢執行計畫產生和編譯最佳化程式碼。編譯的程式碼執行速度更快，因為它消除了使用解譯器的額外負荷。為了將新查詢的延遲降至最低，同時保留編譯程式碼的效能優勢，Amazon Redshift 使用稱為合成的技術。合成會產生預先存在邏輯的輕量型配置，以立即處理新查詢，同時在背景編譯高度最佳化的查詢特定程式碼。這會從查詢執行的關鍵路徑移除編譯。這表示新查詢的啟動速度更快，並提供與後續執行一致的效能。

Amazon Redshift 也會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本機快取在叢集上，並在叢集重新啟動後持續存在的幾乎無限制的遠端快取中。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。透過使用可擴展的編譯服務，Amazon Redshift 會平行編譯程式碼，以提供一致的快速效能。

設計 Amazon Redshift 資料表的最佳實務

本節提供設計資料庫資料表的最佳實務概觀。我們建議您遵循這些最佳實務，以實現最佳的查詢效能和效率。

了解排序索引鍵的運作方式

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。若要有效使用排序索引鍵，建議您執行下列動作：

- 盡可能對資料表進行排序。
- 使用VACUUM排序來還原最佳效能。
- 避免壓縮排序索引鍵資料欄。
- 如果排序索引鍵已壓縮且sortkey1_skew比率顯著較高，則重新建立資料表，而不啟用排序索引鍵的壓縮。
- 避免將函數套用至排序索引鍵資料欄。例如，在以下查詢中，如果您將其轉換為 ，則不會使用trans_dt : TIMESTAMPTZ排序索引鍵資料欄DATE：

```
select order_id, order_amt
from sales
where trans_dt::date = '2021-01-08'::date
```

- 依排序索引鍵順序執行INSERT操作。
- 盡可能在 GROUP BY子句中使用排序索引鍵。

查詢調校秘訣

建議您執行下列動作來調整查詢：

- 一律將複合排序索引鍵從最低基數排列為最高基數，以獲得最佳效果。
- 如果複合排序索引鍵中的前導索引鍵相對是唯一的（也就是具有高基數），則請避免將其他資料欄新增至排序索引鍵。新增其他資料欄對查詢效能的影響很小，但會增加維護成本。

評估排序索引鍵有效性

若要最佳化查詢，您必須能夠評估查詢的有效性。我們建議您使用 SVL_QUERY_SUMMARY 檢視來尋找查詢執行的一般資訊。在此檢視中，您可以使用 屬性IS_RRSCAN來判斷EXPLAIN計劃步驟是否使用範圍限制掃描。您也可以使用 屬性rows_pre_filter來判斷排序索引鍵的選擇性。

您也可以從稱為 v_my_last_query_summary 的 GitHub 使用管理員檢視。檢視會顯示上次執行查詢的資訊。

下列陳述式說明如何尋找有關查詢執行的一般資訊。

```
select lpad(' ',stm+seg+step) || label as label,
       rows,
       bytes,
       is_diskbased,
       is_rrscan,
       rows_pre_filter
from svl_query_summary
where query = pg_last_query_id()
order by stm, seg, step;
```

上述查詢會傳回下列範例輸出。

label	rows	bytes	is_diskbased	is_rrscan	rows_pre_filter
scan tbl=163860 name=orders	1500000	24000000	f	f	1500000
project	1500000	0	f	f	0
project	1500000	0	f	f	0
hash tbl=968	1500000	24000000	f	f	0
scan tbl=163852 name=lineitem	6001215	144029160	f	t	6001215
project	6001215	0	f	f	0
project	6001215	0	f	f	0
hjoin tbl=968	6001215	0	f	f	0
project	6001215	0	f	f	0
project	6001215	0	f	f	0

了解您的資料表

請務必了解資料表的關鍵屬性。若要進一步了解您的資料表，請執行下列動作：

- 使用 PG_TABLE_DEF 來檢視資料表資料欄的相關資訊。

- 使用 SVV_TABLE_INFO 來檢視更完整的資料表相關資訊，包括資料分佈偏斜、金鑰分佈偏斜、資料表大小和統計資料。

選擇正確的資料表分佈樣式

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。選擇資料表分佈樣式的目標是將重新分佈步驟的影響降至最低，方法是在執行查詢之前，將資料放置在需要的位置。

我們建議您使用下列方法來選擇正確的資料表分佈樣式：

- 串連相同節點內的資料列，避免在查詢執行計畫中廣播和重新分佈。例如，透過選取 DISTKEY，您可以在其通用資料欄上分配事實資料表和單維度資料表。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。
- 確定建立分發金鑰的欄沒有扭曲。否則，一個運算節點可以執行比其他運算節點更多的繁重負載。如果您注意到偏斜，請考慮變更分佈索引鍵資料欄。如果資料欄的值是均勻分佈或高基數值，則可以將其視為分佈索引鍵的候選項目。
- 如果聯結條件中使用的資料表很小（小於 1 GB)，請考慮分佈樣式 ALL。
- 您可以壓縮分佈索引鍵，但必須避免壓縮排序索引鍵資料欄（特別是排序索引鍵的第一欄）。

> **Note**
>
> 如果您使用自動資料表最佳化，則不需要選擇資料表的分佈樣式。如需詳細資訊，請參閱《Amazon Redshift 文件》中的使用自動資料表最佳化。若要讓 Amazon Redshift 選擇適當的分佈樣式，請指定分佈樣式的 AUTO。

設計 Amazon Redshift 查詢的最佳實務

本節提供設計查詢的最佳實務概觀。我們建議您遵循本節中的最佳實務，以實現最佳的查詢效能和效率。

避免使用 SELECT * FROM 陳述式

建議您避免使用SELECT * FROM 陳述式。反之，一律列出要分析的資料欄。這可減少查詢執行時間，並掃描 Amazon Redshift Spectrum 查詢的成本。

要避免的內容範例

```
select *
from sales;
```

最佳實務範例

```
select sales_date, sales_amt
from sales;
```

識別查詢問題

建議您檢查 STL_ALERT_EVENT_LOG 檢視，以識別和修正查詢的可能問題。

取得查詢的摘要資訊

我們建議您使用 SVL_QUERY_SUMMARY 和 SVL_QUERY_REPORT 檢視來取得查詢的摘要資訊。您可以使用此資訊來最佳化查詢。

避免交叉聯結

除非絕對必要，否則建議您避免使用跨聯結。如果沒有聯結條件，跨聯結會產生兩個資料表的笛卡兒乘積。跨聯結通常會以巢狀迴圈聯結 （最慢的可能聯結類型） 的形式執行。

要避免的內容範例

```
select c.c_name,
```

```
          n.n_name
from tpch.customer c,
        tpch.nation n;
```

最佳實務範例

```
select c.c_name,
           n.n_name
from tpch.customer c,
join tpch.nation n
  on n.n_nationkey = c.c_nationkey;
```

避免查詢述詞中的函數

建議您避免在查詢述詞中使用 函數。在查詢述詞中使用函數可能會對效能產生負面影響，因為函數通常會為每個資料列增加額外的處理開銷，並減慢查詢的整體執行速度。

要避免的內容範例

```
select sum(o_totalprice)
from tpch.orders
where datepart(year, o_orderdate) = 1992;
```

最佳實務範例

```
select sum(o_totalprice)
from tpch.orders
where o_orderdate between '1992-01-01' and '1992-12-31';
```

避免不必要的轉換

我們建議您避免在查詢上使用不必要的轉換轉換，因為轉換資料類型需要時間和資源，並減慢查詢執行速度。

要避免的內容範例

```
select sum(o_totalprice)
from tpch.orders
where o_ordertime::date = '1992-01-01';
```

最佳實務範例

```
select sum(o_totalprice)
from tpch.orders
where o_ordertime between '1992-01-01 00:00:00' and '1992-12-31 23:59:59';
```

使用 CASE 表達式進行複雜的彙總

我們建議您使用 CASE 表達式來執行複雜的彙總，而不是多次從相同的資料表中選取 。

要避免的內容範例

```
select sum(sales_amt) as us_sales
from sales
where country = 'US';

select sum(sales_amt) as ca_sales
from sales
where country = 'CA';
```

最佳實務範例

```
select sum(case when country = 'US' then sales_amt end) as us_sales,
           sum(case when country = 'CA' then sales_amt end) as ca_sales
from sales;
```

使用子查詢

如果查詢中的一個資料表僅用於述詞條件，且子查詢傳回少量資料列 （ 少於約 200)，建議您使用子查詢。

要避免的內容範例

如果子查詢傳回的資料列少於 200 個：

```
select sum(order_amt) as total_sales
from sales
where region_key IN
        (select region_key
```

```
        from regions
        where state = 'CA');
```

最佳實務範例

如果子查詢傳回大於或等於 200 列：

```
select sum(o.order_amt) as total_sales
from sales o
join regions r
  on r.region_key = o.region_key
  and r.state = 'CA';
```

使用述詞

我們建議您使用述詞來盡可能限制資料集。SQL 中使用述詞來篩選和限制查詢中傳回的資料。透過在述詞中指定條件，您可以指定哪些資料列必須根據指定的條件包含在查詢結果中。這可讓您僅擷取您感興趣的資料，並改善查詢的效率和準確性。如需詳細資訊，請參閱 Amazon Redshift 文件中的條件。

新增述詞以使用聯結篩選資料表

我們建議您新增述詞來篩選參與聯結的資料表，即使述詞套用相同的篩選條件。使用述詞篩選具有 SQL 聯結的資料表，可透過減少必須處理的資料量，以及減少中繼結果集的大小，來改善查詢效能。透過在 WHERE子句中指定聯結操作的條件，查詢執行引擎可以在聯結之前消除不符合條件的資料列。這會導致較小的結果集和更快的查詢執行。

要避免的內容範例

```
select p.product_name, sum(o.order_amt)
from sales o
join product p
   on r.product_key = o.product_key
where o.order_date > '2022-01-01';
```

最佳實務範例

```
select p.product_name, sum(o.order_amt)
from sales o
join product p
```

```
  on p.product_key = o.product_key
  and p.added_date > '2022-01-01'
where o.order_date > '2022-01-01';
```

針對述詞使用最便宜的運算子

在述詞中，使用您可以花費最低的運算子。比較條件運算子偏好 LIKE 運算子。LIKE運算子仍偏好 SIMILAR TO 或 POSIX 運算子。

在 GROUP BY 子句中使用排序索引鍵

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清單僅包含排序索引鍵資料欄時，查詢可能符合單階段彙總的資格，其中一個也是分佈索引鍵。GROUP BY 清單中的排序索引鍵資料欄必須包含第一個排序索引鍵，後面接著您要依排序索引鍵順序使用的其他排序索引鍵。

利用具體化視觀表

如果可能，將複雜程式碼取代為具體化視觀表來重寫查詢，這將大幅改善查詢的效能。如需詳細資訊，請參閱《Amazon Redshift 文件》中的在 Amazon Redshift 中建立具體化視觀表。

請注意 GROUP BY 和 ORDER BY 子句中的資料欄

如果您同時使用 GROUP BY和 ORDER BY子句，請確定您在 GROUP BY和 ORDER BY子句中以相同的順序放置資料欄。 GROUP BY 隱含地需要對資料進行排序。如果您的ORDER BY子句不同，則資料必須排序兩次。

要避免的內容範例

```
select a, b, c, sum(d)
from a_table
group by b, c, a
order by a, b, c
```

最佳實務範例

```
select a, b, c, sum(d)
```

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

使用 Amazon Redshift Spectrum 的最佳實務

本節提供使用 Amazon Redshift Spectrum 的最佳實務概觀。當您使用 Redshift Spectrum 時，我們建議您遵循這些最佳實務來達到最佳效能：

- 假設檔案類型對 Redshift Spectrum 查詢效能有重大影響。若要改善效能，請使用 ORC 或 Parquet 等單欄式編碼檔案，並僅針對非常小的維度資料表使用 CSV 格式。
- 使用字首型分割來利用分割區剔除。這表示使用輸入資料湖中分割區的篩選條件。
- Redshift Spectrum 會自動擴展以處理大型請求，因此盡可能在 Redshift Spectrum 中執行 （例如述詞下推）。
- 請注意經常篩選資料欄上的分割區檔案。如果資料由一或多個篩選的資料欄分割，Redshift Spectrum 可以利用分割區剔除並略過掃描不需要的分割區和檔案。常見做法是根據時間對資料進行分割。
- 您可以使用下列查詢，檢查分割區的有效性和 Redshift Spectrum 查詢的效率。

```
Select query,
           segment,
           max(assigned_partitions) as total_partitions,
           max(qualified_partitions) as qualified_partitions
From svl_s3partition
Where query=pg_last_query_id()
Group by 1,2;
```

上述查詢顯示下列項目：

- total_partitions – 所辨識的分割區數量 AWS Glue Data Catalog
- qualified_partitions – 針對 Redshift Spectrum 查詢存取的 Amazon Simple Storage Service (Amazon S3) 字首數量

- 您也可以檢查SVL_S3QUERY_SUMMARY系統資料表，以了解分割區的有效性和 Redshift Spectrum 查詢的效率。若要這麼做，請使用下列陳述式。

```
Select *
From svl_s3query_summary
Where query=pg_last_query_id();
```

除了顯示分割區剔除效率的檔案之外is_partitioned，上述查詢還會傳回更多資訊，包括 s3_scanned_rows/bytes、 和 s3_returned_rows/bytes值。

Redshift Spectrum 中的述詞下推

使用述詞下推可避免在 Amazon Redshift 叢集中耗用資源。您可以將許多 SQL 操作向下推送至 Redshift Spectrum layer。我們建議您盡可能利用這項功能。

請謹記以下幾點：

- 您可以在 Redshift Spectrum layer 中完全評估某些類型的 SQL 操作，包括下列項目：
  - GROUP BY 子句
  - 比較和模式比對條件 （ 例如 LIKE)
  - 彙總函數 （ 例如 COUNT、、MIN、 SUM AVG和 MAX)
  - regex_replace、date_trunc、 to_upper和其他 函數
- 您無法將某些操作推送至 Redshift Spectrum layer，包括 DISTINCT和 ORDER BY。請盡可能ORDER BY在查詢的最上層 執行，因為排序是在領導節點中完成。
- 檢查您的查詢EXPLAIN計劃，以驗證述詞下推是否有效。若要在 EXPLAIN命令中尋找 Redshift Spectrum 部分，請查看以下步驟：
  - S3 循序掃描
  - S3 HashAggregate
  - S3 查詢掃描
  - 循序掃描 PartitionInfo (分割區資訊)
  - 分割區循環
- 在查詢中使用最少數量的資料欄。如果資料是 Parquet 或 ORC 格式，Redshift Spectrum 可以消除掃描的資料欄。
- 廣泛使用分割區進行平行處理和消除分割區，並盡可能將檔案大小保持在至少 64 MB。
- TABLE PROPERTIES 'numRows'='nnn' 如果您使用 CREATE EXTERNAL TABLE或 ，請設定 ALTER TABLE。Amazon Redshift 不會分析外部資料表，以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。如果未設定統計資料，Amazon Redshift 會假設外部資料表是較大的資料表，而本機資料表是較小的資料表。

Redshift Spectrum 的查詢調校秘訣

調整查詢時，建議您謹記下列事項：

- Amazon Redshift 叢集可以針對查詢進行的 Redshift Spectrum 節點數目，會與叢集中的配量數目綁定。
- 調整叢集大小可以讓叢集的本機運算設定檔、儲存設定檔和 Amazon S3 資料湖查詢的查詢功能受益。
- 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。
- 由於 Redshift Spectrum 會自動擴展以處理大型請求，因此每當您將處理推送到 Redshift Spectrum layer 時，整體效能都會提高。

資源

- Amazon Redshift 最佳實務 (Amazon Redshift 文件)
- Amazon Redshift 設計查詢的最佳實務 (Amazon Redshift 文件)
- 調校查詢效能 (Amazon Redshift 文件)
- 查詢計劃 (Amazon Redshift 文件)
- 改善 Amazon Redshift Spectrum 查詢效能 (Amazon Redshift 文件)
- 了解 Amazon Redshift 中的查詢生命週期 (AWS 方案指引)

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
已移除 AQUA	我們移除了進階查詢加速器 (AQUA) 的相關資訊。	2024 年 6 月 14 日
初次出版	—	2023 年 2 月 3 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。