aws

ユーザーガイド

AWS Schema Conversion Tool

Version 1.0.672

AWS Schema Conversion Tool: ユーザーガイド

Table of Contents

AWS Schema Conversion Tool とは

AWS Schema Conversion Tool (AWS SCT) を使用すると、データベースエンジン間で既存のデータベーススキーマを変換できます。リレーショナル OLTP スキーマやデータウェアハウススキーマを変換できます。変換されたスキーマは、Amazon Relational Database Service (Amazon RDS) MySQL、MariaDB、Oracle、SQL Server、PostgreSQL DB、Amazon Aurora DB クラスター、または Amazon Redshift クラスターに適しています。変換されたスキーマは、Amazon EC2 インスタンスでデータベースと共に使用するか、Amazon S3 バケットでデータとして保存することもできます。

AWS SCT は、Amazon S3 バケットや他の AWS リソースに接続するときに、連邦情報処理標準 (FIPS) など、いくつかの業界標準をサポートします。AWS SCT は、米国連邦政府のリスク認証管理プログラム (FedRAMP) にも準拠しています。AWS およびコンプライアンスの取り組みの詳細については、コンプライアンスプログラムによる AWS 対象範囲内のサービスを参照してください。

AWS SCT では、以下の OLTP 変換がサポートされています。

ソースデータベース	ターゲットデータベース
IBM Db2 for z/OS (バージョン 12)	Amazon Aurora MySQL 互換エディション (Aurora MySQL)、Amazon Aurora PostgreSQL 互換エディション (Aurora PostgreSQL)、MySQL, PostgreSQL 詳細については、「IBM Db2 for z/OS をソースとして使用する」を参照してください。
IBM Db2 LUW (バージョン 9.1、9.5、9.7、10.5、11.1、および 11.5)	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL 詳細については、「IBM Db2 LUW をソースとして使用する」を参照してください。
Microsoft Azure SQL データベース	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL 詳細については、「Azure SQL データベースのソースとしての使用」を参照してください。

ソースデータベース	ターゲットデータベース
Microsoft SQL Server (バージョン 2008 R2、2012、2014、2016、2017、2019、2022)	Aurora MySQL、Aurora PostgreSQL、Babelfish for Aurora PostgreSQL (評価レポートのみ)、MariaDB、Microsoft SQL Server、MySQL、PostgreSQL 詳細については、「SQL Server のソースとしての使用」を参照してください。
MySQL (バージョン 5.5 以上)	Aurora PostgreSQL、MySQL、PostgreSQL 詳細については、「ソースとしての MySQL の使用」を参照してください。 AWS SCT を使用せずに、MySQL から Aurora MySQL DB クラスターにスキーマとデータを移行できます。詳細については、「Amazon Aurora DB クラスターへのデータの移行」を参照してください。
Oracle (バージョン 10.1 以降)	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、Oracle、PostgreSQL 詳細については、「ソースとしての Oracle Database の使用」を参照してください。
PostgreSQL (バージョン 9.1 以降)	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL 詳細については、「PostgreSQL のソースとしての使用」を参照してください。
SAP ASE (バージョン 12.5.4、15.0.2、15.5、15.7、16.0)	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL 詳細については、「ソースとしての SAP ASE (Sybase ASE) の使用」を参照してください。

AWS SCT では、以下のデータウェアハウスの変換がサポートされています。

ソースデータウェアハウス	ターゲットデータウェアハウス
Amazon Redshift	Amazon Redshift 詳細については、「ソースとしての Amazon Redshift の使用」を参照してください。
Azure Synapse Analytics	Amazon Redshift 詳細については、「ソースとしての Azure Synapse Analytics の使用」を参照してください。
BigQuery	Amazon Redshift 詳細については、「ソースとして BigQuery を使用する」を参照してください。
Greenplum データベース (バージョン 4.3 および 6.21 以降)	Amazon Redshift 詳細については、「ソースとしての Greenplum データベースの使用」を参照してください。
Microsoft SQL Server (バージョン 2008 以降)	Amazon Redshift 詳細については、「SQL Server データウェアハウスをソースとしての使用」を参照してください。
Netezza (バージョン 7.0.3 以降)	Amazon Redshift 詳細については、「ソースとしての Netezza の使用」を参照してください。
Oracle (バージョン 10.1 以降)	Amazon Redshift

ソースデータウェアハウス	ターゲットデータウェアハウス
	詳細については、「ソースとしての Oracle データウェアハウスの使用」を参照してください。
Snowflake (バージョン 3)	Amazon Redshift 詳細については、「ソースとしての Snowflake の使用」を参照してください。
Teradata (バージョン 13 以降)	Amazon Redshift 詳細については、「ソースとしての Teradata の使用」を参照してください。
Vertica (バージョン 7.2.2 以降)	Amazon Redshift 詳細については、「ソースとしての Vertica の使用」を参照してください。

AWS SCT では、以下のデータ NoSQL データベース変換がサポートされています。

ソースデータベース	ターゲットデータベース
Apache Cassandra (バージョン 2.1.x、2.2.16、3.11.x)	Amazon DynamoDB 詳細については、「Apache Cassandra をソースとしての使用」を参照してください。

AWS SCT は、次の抽出、変換、ロード (ETL) プロセスの変換をサポートします。詳細については、「ETL プロセスの変換」を参照してください。

ソース	ターゲット
Informatica ETL スクリプト	Informatica

ソース	ターゲット
Microsoft SQL Server Integration Services (SSIS) ETL パッケージ	AWS Glue、または AWS Glue Studio
Teradata Basic Teradata Query (BTEQ) からのコマンドが埋め込まれたシェルスクリプト	Amazon Redshift RSQL
Teradata BTEQ ETL スクリプト	AWS Glue または Amazon Redshift RSQL
Teradata FastExport ジョブスクリプト	Amazon Redshift RSQL
Teradata FastLoad ジョブスクリプト	Amazon Redshift RSQL
Teradata Mulload ジョブスクリプト	Amazon Redshift RSQL

AWS SCTは、以下のビッグデータフレームワークの移行をサポートします。詳細については、「ビッグデータフレームワークの移行」を参照してください。

ソース	ターゲット
Apache Hive (バージョン 0.13.0 以降)	Amazon EMR 上の Hive
Apache HDFS	Amazon EMR 上の Amazon S3 または HDFS
Apache Oozie	AWS Step Functions

スキーマ変換の概要

AWS SCT には、ソースデータベースのデータベーススキーマをターゲットとなる Amazon RDS インスタンスと互換性のある形式に自動変換するための、プロジェクトベースのユーザーインターフェイスが用意されています。ソースデータベースのスキーマを自動変換できない場合、AWS SCT では、ターゲットとなる Amazon RDS データベースで同等のスキーマを作成する方法についてのガイダンスが表示されます。

AWS SCT をインストールする方法の詳細については、「のインストール、検証、更新 AWS SCT」を参照してください。

AWS SCT ユーザーインターフェイスの概要については、「AWS SCT ユーザーインターフェイスの使用」を参照してください。

変換プロセスの詳細については、「AWS SCT を使用したデータベーススキーマの変換」を参照してください。

AWS SCT には、1 つのデータベースエンジンから別のデータベースエンジンに既存のデータベーススキーマを変換する以外にも、データやアプリケーションを AWS クラウドに移動するのに役立つ追加機能があります。

- データ抽出エージェントを使用して、Amazon Redshift への移行を準備するためにデータウェアハウスからデータを抽出できます。データ抽出エージェントを管理するには、AWS SCT を使用できます。詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。
- AWS SCT を使用して、AWS DMS エンドポイントやタスクを作成できます。AWS SCT からこれらのタスクを実行およびモニタリングできます。詳細については、「AWS SCT の AWS DMS との併用」を参照してください。
- 場合によっては、データベース機能で同等の Amazon RDS または Amazon Redshift 機能に変換できないことがあります。AWS SCT 拡張パックウィザードを使用すると、AWS Lambda 関数と Python ライブラリをインストールし、変換できない機能をエミュレートできます。詳細については、「AWS SCT 拡張パックの使用」を参照してください。
- 既存の Amazon Redshift データベースを最適化するために AWS SCT を使用できます。AWS SCT では、ソートキーとディストリビューションキーでデータベースを最適化することをお勧めします。詳細については、「AWS SCT を使用した Amazon Redshift の最適化」を参照してください。
- AWS SCT を使用して、同エンジンを実行する Amazon RDS DB インスタンスに既存のオンプレミスのデータベーススキーマをコピーできます。この機能を使用して、クラウドへの移動やライセンスタイプの変更にかかるコスト削減の可能性を分析できます。
- AWS SCT を使用して、C++、C#、Java などのアプリケーションコードで SQL を変換できます。変換された SQL コードは表示、分析、編集、保存できます。詳細については、「AWS SCT を使用したアプリケーション SQL の変換」を参照してください。
- AWS SCT を使用して、抽出、変換、ロード (ETL) プロセスを移行できます。詳細については、「AWS Schema Conversion Tool を使用した抽出、変換、ロード (ETL) プロセスの変換」を参照してください。

フィードバックの提供

AWS SCT に関するフィードバックを提供できます。バグレポートの提出、機能リクエストの提出、一般情報の提供ができます。

AWS SCT に関するフィードバックを提供するには

1. AWS Schema Conversion Tool を開始します。
2. [Help] (ヘルプ) メニューを開き、[Leave Feedback] (フィードバックを残す) を選択します。[Leave Feedback] (フィードバックを残す) ダイアログボックスが表示されます。
3. [Area] (エリア) で、[Information] (情報)、[Bug report] (バグレポート)、または [Feature request] (機能リクエスト) を選択します。
4. [Source database] (ソースデータベース) で、ソースデータベースを選択します。フィードバックが特定のデータベースに特有でない場合、[Any] (すべて) を選択します。
5. [Target database] (ターゲットデータベース) で、ターゲットデータベースを選択します。フィードバックが特定のデータベースに特有でない場合、[Any] (すべて) を選択します。
6. [Title] タイトル() で、フィードバックのタイトルを入力します。
7. [Message] (メッセージ) で、フィードバックを入力します。
8. [Send] (送信) を選択して、フィードバックを送信します。

のインストール、検証、更新 AWS SCT

AWS Schema Conversion Tool（ AWS SCT) は、プロジェクトベースのユーザーインターフェイスを提供するスタンドアロンアプリケーションです。 AWS SCT は Microsoft Windows、Fedora Linux、Ubuntu Linux で使用できます。 AWS SCT は 64 ビットオペレーティングシステムでのみサポートされています。

ディス AWS SCT トリビューションファイルの正しいバージョンを取得するには、圧縮ファイルをダウンロードした後に検証手順を提供します。この手順を使用してファイルを検証できます。

AWS SCT は、スタンドアロンアプリケーションとコマンドラインツールの両方として使用できます。コマンドラインツールの詳細については、「」を参照してくださいAWS SCT CLI。

トピック

のインストール AWS SCT

は、次のオペレーティングシステム AWS SCT にインストールできます。

- Microsoft Windows 10
- Fedora Linux 36 以降
- Ubuntu Linux 18 以降

をインストールするには AWS SCT

1. オペレーティングシステムのリンクを使用して、 AWS SCT インストーラを含む圧縮ファイルをダウンロードします。圧縮されたファイルの拡張子はすべて .zip です。 AWS SCT インストーラファイルを抽出すると、オペレーティングシステムに適した形式になります。

    - Microsoft Windows

- Ubuntu Linux (.deb)
- Fedora Linux (.rpm)

2. 以下に示すように、オペレーティングシステムの AWS SCT インストーラファイルを抽出します。

オペレーティングシステム	ファイル名
Fedora Linux	`aws-schema-conversion-tool-1.0.` *`build-number`* `.x86_64.rpm`
Microsoft Windows	`AWS Schema Conversion Tool-1.0.` *`build-number`* `.msi`
Ubuntu Linux	`aws-schema-conversion-tool-1.0.` *`build-number`* `.deb`

3. 前のステップで抽出した AWS SCT インストーラファイルを実行します。次に示すように、オペレーティングシステムの手順を使用します。

オペレーティングシステム	インストール手順
Fedora Linux	ファイルをダウンロードしたフォルダーで、次のコマンドを実行します。 `sudo yum install aws-schema-conversion-tool-1.0.` *`build-number`* `.x86_64.rpm`
Microsoft Windows	ファイルをダブルクリックしてインストーラを実行します。
Ubuntu Linux	ファイルをダウンロードしたフォルダーで、次のコマンドを実行します。 `sudo dpkg -i aws-schema-conversion-tool-1.0.` *`build-number`* `.deb`

4. ソースとターゲットデータベースエンジン用 Java Database Connectivity (JDBC) ドライバをダウンロードします。手順とダウンロードのリンクについては、「必要なデータベースドライバーのダウンロード」を参照してください。

これで、 AWS SCT アプリケーションのセットアップが完了しました。アプリケーションのアイコンをダブルクリックして AWS SCTを実行します。

AWS SCT ファイルのダウンロードの検証

のディストリビューションファイルを検証する方法はいくつかあります AWS SCT。最も簡単な方法としては、ファイルのチェックサムを AWSの公表チェックサムと比較します。追加のセキュリティレベルとして、ファイルのインストール先のオペレーティングシステムに応じて、次の手順を使用してディストリビューションを検証できます。

このセクションでは、次のトピックについて説明します。

トピック

AWS SCT ファイルのチェックサムの検証

AWS SCT 圧縮ファイルをダウンロードまたは保存するときに発生した可能性のあるエラーを検出するために、ファイルのチェックサムを によって提供された値と比較できます AWS。 はチェックサムに SHA256 アルゴリズム AWS を使用します。

チェックサムを使用して AWS SCT ディストリビューションファイルを検証するには

1. インストールセクションのリンクを使用してディス AWS SCT トリビューションファイルをダウンロードします。詳細については、「のインストール AWS SCT」を参照してください。
2. 最新のチェックサムファイル (sha256Check.txt) をダウンロードします。このファイルには、AWS SCT 最新バージョンのチェックサムが含まれています。例えば、このファイルは次のように表示されます。

```
Fedora   b4f5f66f91bfcc1b312e2827e960691c269a9002cd1371cf1841593f88cbb5e6
Ubuntu   4315eb666449d4fcd95932351f00399adb6c6cf64b9f30adda2eec903c54eca4
Windows  6e29679a3c53c5396a06d8d50f308981e4ec34bd0acd608874470700a0ae9a23
```

3. ディストリビューションファイルが含まれているディレクトリで、オペレーティングシステムの SHA256 検証コマンドを実行します。例えば、Linux で以下のコマンドを実行します。

```
shasum -a 256 aws-schema-conversion-tool-1.0.latest.zip
```

4. コマンドの結果と sha256Check.txt ファイルに表示されている値を比較します。チェックサムが一致する場合は、配布ファイルを実行しても安全です。チェックサムが一致しない場合は、配布ファイルを実行せず、AWS サポートに連絡してください。

Fedora での AWS SCT RPM ファイルの検証

AWS は、ディストリビューションファイルのチェックサムに加えて、別のレベルの検証を提供します。ディストリビューションファイル内のすべての RPM ファイルは、 AWS プライベートキーによって署名されます。パブリック GPG キーは、amazon.com.public.gpg-key で表示できます。

Fedora で AWS SCT RPM ファイルを確認するには

1. インストールセクションのリンクを使用してディス AWS SCT トリビューションファイルをダウンロードします。
2. AWS SCT ディストリビューションファイルのチェックサムを確認します。
3. ディストリビューションファイルの内容を抽出します。検証する RPM ファイルを見つけます。
4. amazon.com.public.gpg-key から GPG パブリックキーをダウンロードします
5. 次のコマンドを使用して、パブリックキーを RPM DB にインポートします (適切なアクセス許可があることを確認してください)。

```
sudo rpm --import aws-dms-team@amazon.com.public.gpg-key
```

6. 次のコマンドを実行してインポートが成功したことを確認します。

```
rpm -q --qf "%{NAME}-%{VERSION}-%{RELEASE} \n %{SUMMARY} \n" gpg-pubkey-
ea22abf4-5a21d30c
```

7. 次のコマンドを実行して RPM 署名を確認します。

```
rpm --checksig -v aws-schema-conversion-tool-1.0.build number-1.x86_64.rpm
```

Ubuntu での AWS SCT DEB ファイルの検証

AWS は、ディストリビューションファイルのチェックサムに加えて、別のレベルの検証を提供します。ディストリビューションファイル内のすべての DEB ファイルは、GPG デタッチ済み署名で署名されます。

Ubuntu で AWS SCT DEB ファイルを確認するには

1. インストールセクションのリンクを使用してディス AWS SCT トリビューションファイルをダウンロードします。
2. AWS SCT ディストリビューションファイルのチェックサムの検証。
3. ディストリビューションファイルの内容を抽出します。検証する DEB ファイルを見つけます。
4. デタッチされた署名を aws-schema-conversion-tool-1.0.latest.deb.asc からダウンロードします。
5. amazon.com.public.gpg-key から GPG パブリックキーをダウンロードします。
6. 次のコマンドを実行して GPG パブリックキーをインポートします。

```
gpg --import aws-dms-team@amazon.com.public.gpg-key
```

7. 次のコマンドを実行して署名を検証します。

```
gpg --verify aws-schema-conversion-tool-1.0.latest.deb.asc aws-schema-conversion-
tool-1.0.build number.deb
```

Microsoft Windows での AWS SCT MSI ファイルの検証

AWS は、ディストリビューションファイルのチェックサムに加えて、別のレベルの検証を提供します。MSI ファイルにはデジタル署名があり、 によって署名されたことを確認できます AWS。

Windows で AWS SCT MSI ファイルを確認するには

1. インストールセクションのリンクを使用してディス AWS SCT トリビューションファイルをダウンロードします。
2. AWS SCT ディストリビューションファイルのチェックサムの検証。
3. ディストリビューションファイルの内容を抽出します。検証する MSI ファイルを見つけます。

4. Windows エクスプローラーで、MSI ファイルを右クリックして [Properties] (プロパティ) を選択します。
5. [Digital Signatures] (デジタル署名) タブを選択します。
6. デジタル署名が Amazon Services LLC によるものであることを確認します。

必要なデータベースドライバーのダウンロード

が正しく動作 AWS SCT するには、ソースデータベースエンジンとターゲットデータベースエンジンの JDBC ドライバーをダウンロードします。仮想ターゲットデータベースプラットフォームを使用する場合、ターゲットデータベースエンジンの JDBC ドライバーをダウンロードする必要はありません。詳細については、「仮想ターゲットの使用」を参照してください。

ドライバをダウンロードしたら、ドライバファイルの場所を指定します。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

次の場所から、データベースドライバをダウンロードできます。

> **Important**
>
> 使用できるドライバの最新バージョンをダウンロードします。次の表に、 でサポートされているデータベースドライバーの最小バージョンを示します AWS SCT。

データベースエンジン	ドライバ	ダウンロード場所
Amazon Aurora MySQL 互換エディション	`mysql-connector-java-5.1.6.jar`	https://www.mysql.com/products/connector/
Amazon Aurora PostgreSQL 互換エディション	`postgresql-42.2.19.jar`	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar

データベースエンジン	ドライバ	ダウンロード場所
Amazon EMR	`HiveJDBC42.jar`	http://awssupportdatasvcs.com/bootstrap-actions/Simba/latest/
Amazon Redshift	`redshift-jdbc42-2.1.0.9.jar`	https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.9/redshift-jdbc42-2.1.0.9.zip
Amazon Redshift Serverless	`redshift-jdbc42-2.1.0.9.jar`	https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.9/redshift-jdbc42-2.1.0.9.zip
Apache Hive	`hive-jdbc-2.3.4-standalone.jar`	https://repo1.maven.org/maven2/org/apache/hive/hive-jdbc/2.3.4/hive-jdbc-2.3.4-standalone.jar
Azure SQL データベース	`mssql-jdbc-7.2.2.jre11.jar`	https://docs.microsoft.com/en-us/sql/connect/jdbc/release-notes-for-the-jdbc-driver?view=sql-server-ver15#72
Azure Synapse Analytics	`mssql-jdbc-7.2.2.jre11.jar`	https://docs.microsoft.com/en-us/sql/connect/jdbc/release-notes-for-the-jdbc-driver?view=sql-server-ver15#72
Greenplum データベース	`postgresql-42.2.19.jar`	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar
IBM Db2 for z/OS	`db2jcc-db2jcc4.jar`	https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads-db2-zos
IBM Db2 LUW	`db2jcc-db2jcc4.jar`	https://www.ibm.com/support/pages/node/382667
MariaDB	`mariadb-java-client-2.4.1.jar`	https://downloads.mariadb.com/Connectors/java/connector-java-2.4.1/mariadb-java-client-2.4.1.jar

データベースエンジン	ドライバ	ダウンロード場所
Microsoft SQL Server	`mssql-jdbc-10.2.jar`	https://docs.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15
MySQL	`mysql-connector-java-8.0.15.jar`	https://dev.mysql.com/downloads/connector/j/
Netezza	`nzjdbc.jar` クライアントツールのソフトウェアを使用します。ドライババージョン 7.2.1 をダウンロードします。これはデータウェアハウスバージョン 7.2.0 と後方互換性があります。	http://www.ibm.com/support/knowledgecenter/SSULQD_7.2.1/com.ibm.nz.datacon.doc/c_datacon_plg_overview.html
Oracle	`ojdbc8.jar` ドライバーバージョン 8 以降がサポートされます。	https://www.oracle.com/database/technologies/jdbc-ucp-122-downloads.html
PostgreSQL	`postgresql-42.2.19.jar`	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar
SAP ASE (Sybase ASE)	`jconn4.jar`	jConnect JDBC ドライバー
Snowflake	`snowflake-jdbc-3.9.2.jar` 詳細については、「JDBC ドライバーのダウンロード/統合」を参照してください。	https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/3.9.2/snowflake-jdbc-3.9.2.jar

データベースエンジン	ドライバ	ダウンロード場所

Teradata	`terajdbc4.jar`
	
	`tdgssconfig.jar`
	
	Teradata JDBC ドライバーバージョン 16.20.00.11 以降では、このファイルは必要ありません。`tdgssconfig.jar`
		https://downloads.teradata.com/download/connectivity/jdbc-driver

Vertica	`vertica-jdbc-9.1.1-0.jar` ドライバーバージョン 7.2.0 以降がサポートされます。	https://www.vertica.com/client_drivers/9.1.x/9.1.1-0/vertica-jdbc-9.1.1-0.jar

Linux での JDBC ドライバのインストール

次の手順を使用して、 で使用する JDBC ドライバーを Linux システムにインストールできます AWS SCT。

JDBC ドライバを Linux システムにインストールするには

1. JDBC ドライバを保存するディレクトリを作成します。

```
PROMPT>sudo mkdir -p /usr/local/jdbc-drivers
```

2. 次に示すコマンドを使用して、データベースエンジン用の JDBC ドライバをインストールします。

データベースエンジン	インストールコマンド
Amazon Aurora (MySQL との互換性あり)	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz`
Amazon Aurora (PostgreSQL との互換性あり)	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo cp -a /tmp/postgresql-X.X.X.jre7.tar .`
Microsoft SQL Server	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/sqljdbc_X.X.X_enu.tar.gz`
MySQL	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz`

Oracle	`PROMPT> cd /usr/local/jdbc-drivers`
	`PROMPT> sudo mkdir oracle-jdbc`
	`PROMPT> cd oracle-jdbc`
	`PROMPT> sudo cp  -a /tmp/ojdbc8.jar .`

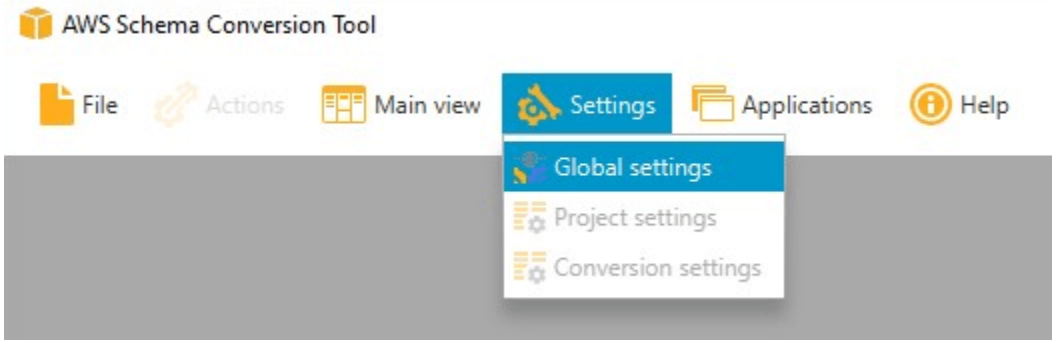
PostgreSQL	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo cp -a /tmp/postgresql-X.X.X.jre7.tar .`

グローバル設定でのドライバパスの保存

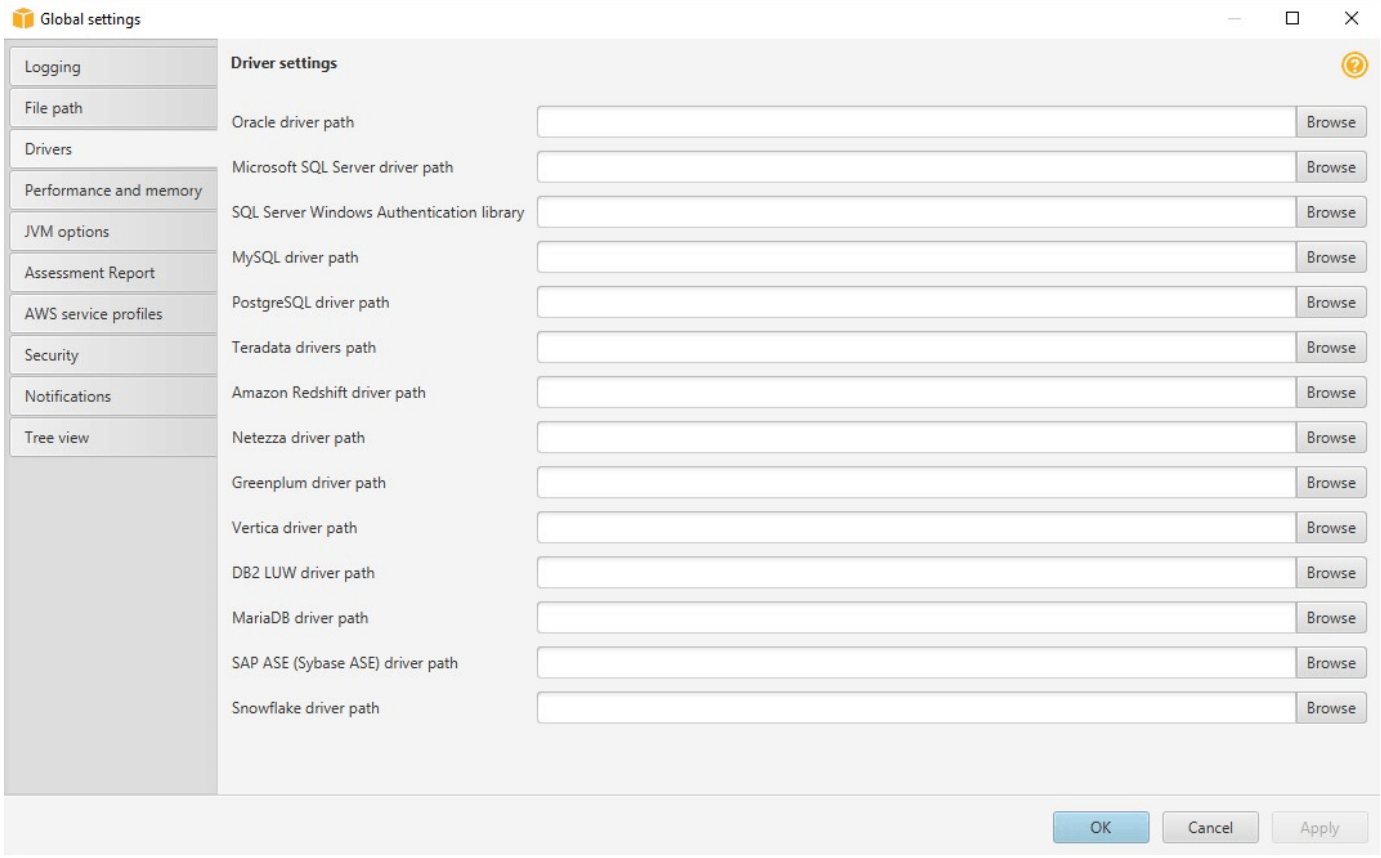
必要な JDBC ドライバーをダウンロードしてインストールしたら、設定でドライバーの場所をグローバルに設定できます AWS SCT 。ドライバの場所をグローバルに設定しない場合、データベースに接続する際にドライバの場所が尋ねられます。

ドライバファイルの場所を更新するには

1. で AWS SCT、設定 を選択し、グローバル設定 を選択します。



2. [Global settings] (グローバル設定) で、[Drivers] (ドライバー) を選択します。ソースデータベースエンジンとターゲット Amazon RDS DB インスタンスデータベースエンジン用の JDBC ドライバへのファイルパスを追加します。



3. ドライバのパスを追加し終えたら、[OK] を選択します。

更新中 AWS SCT

AWS は、 の新機能 AWS SCT で定期的に更新します。以前のバージョンから更新する場合は、新しい AWS SCT プロジェクトを作成し、使用しているデータベースオブジェクトを再構築します。

の更新が存在するかどうかを確認できます AWS SCT。

の更新を確認するには AWS SCT

1. で、ヘルプ を選択し AWS SCT、更新の確認 を選択します。
2. [Check for Updates] (更新プログラムを確認) ダイアログボックスで、[What's New] (最新情報) を選択します。リンクが表示されない場合は、最新バージョンを使用しています。

AWS SCT CLI

コマンドライン用に AWS SCT CLI をダウンロードできます。JAR をダウンロードするには、次のリンクを使用します。

AWSSchemaConversionToolBatch.jar

AWS SCT ユーザーインターフェイスの使用

以下のトピックは、AWS SCT ユーザーインターフェイスの操作に役立ちます。AWS SCT のインストールの詳細については、「のインストール、検証、更新 AWS SCT」を参照してください。

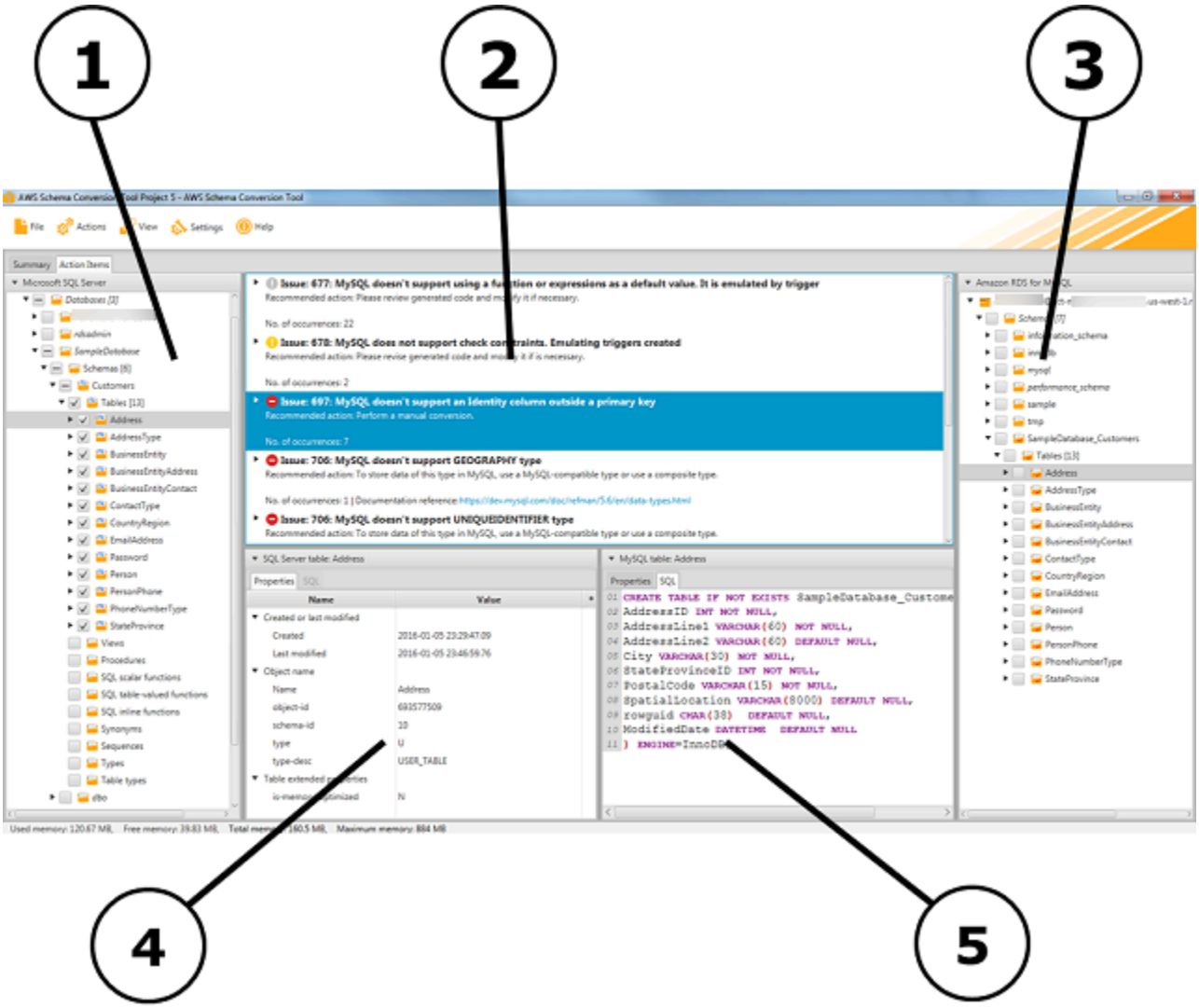
トピック

AWS SCT プロジェクトウィンドウ

以下の図に示しているのは、スキーマ移行プロジェクトを作成してスキーマを変換するときに使用する、AWS SCT のセクションです。

1. 左のパネルでは、ソースデータベースのスキーマがツリービューで表示されます。データベーススキーマは "遅延ロード" されます。つまり、ツリービューから項目を選択すると、AWS SCT では、ソースデータベースの現在のスキーマが取得されて表示されます。

2. 上中央のパネルでは、ターゲットデータベースエンジンに自動変換できなかったソースデータベースエンジンのスキーマ要素について、アクション項目が表示されます。
3. 右のパネルでは、ターゲット DB インスタンスのスキーマがツリービューで表示されます。データベーススキーマは "遅延ロード" されます。つまり、ツリービューから項目を選択した時点で、AWS SCT では、ターゲットデータベースの現在のスキーマが取得されて表示されます。



4. 左下のパネルでは、スキーマ要素を選択すると、プロパティが表示されます。このプロパティには、ソーススキーマ要素と、ソースデータベースでその要素を作成するための SQL コマンドが記述されています。
5. 右下のパネルでは、スキーマ要素を選択すると、プロパティが表示されます。このプロパティには、ターゲットスキーマ要素と、ターゲットデータベースでその要素を作成するための SQL コマ

ンドが記述されています。この SQL コマンドを編集し、更新したコマンドをプロジェクトで保存できます。

AWS SCT の起動

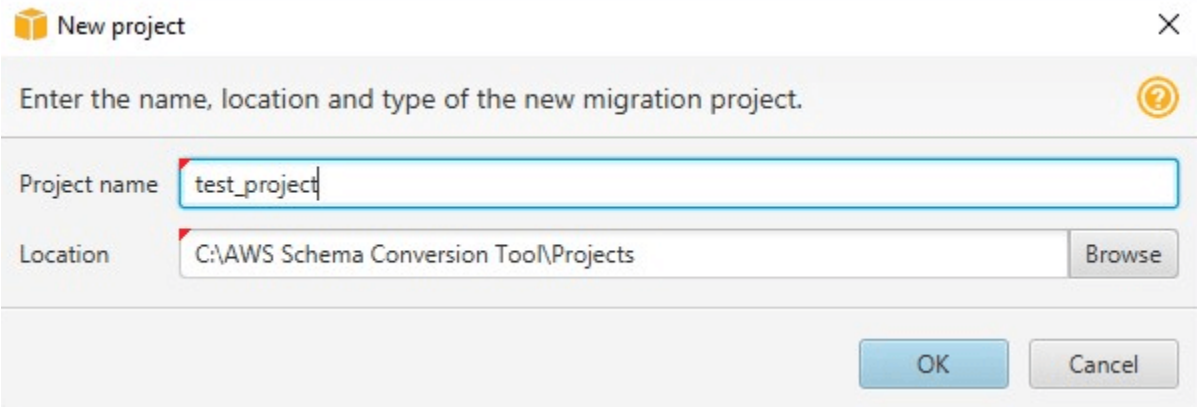
AWS Schema Conversion Tool を起動するには、アプリケーションアイコンをダブルクリックします。

AWS SCT プロジェクトの作成

以下の手順に従って AWS Schema Conversion Tool プロジェクトを作成します。

プロジェクトを作成するには

1. AWS Schema Conversion Tool を開始します。
2. [ファイル] メニューで [新しいプロジェクト] を選択します。[新しいプロジェクト] ダイアログボックスが表示されます。



3. コンピュータにローカルに保存されているプロジェクトの名前を入力します。
4. ローカルプロジェクトファイルの場所を入力します。
5. OK を選択して、AWS SCT プロジェクトを作成します。
6. [Add source] (ソースの追加) を選択して、新しいソースデータベースをAWS SCT プロジェクトに追加します。AWS SCT プロジェクトには、複数のソースデータベースを追加できます。
7. [Add target] (ターゲットの追加) を選択して、新しいターゲットプラットフォームを AWS SCT プロジェクトに追加します。AWS SCT プロジェクトには、複数のターゲットプラットフォームを追加できます。
8. 左側のパネルでソースデータベーススキーマを選択します。

9. 右側のパネルでは、選択したソーススキーマのターゲットデータベースプラットフォームを指定します。
10. [Create mapping] (マッピングの作成) を選択します。このボタンは、ソースデータベーススキーマとターゲットデータベースプラットフォームを選択するとアクティブになります。詳細については、「マッピングルールの作成」を参照してください。

これで、AWS SCT プロジェクトが設定されました。プロジェクトの保存、データベース移行評価レポートの作成、ソースデータベーススキーマの変換を行うことができます。

AWS SCT での新しいプロジェクトウィザードの使用

「新規プロジェクト」ウィザードを使用して、新しいデータベース移行プロジェクトを作成できます。このウィザードは、移行ターゲットを決定したり、データベースに接続したりするのに役立ちます。サポートされているすべてのターゲット宛先について、移行がどれほど複雑になるかを推定します。ウィザードを実行すると、AWS SCT でデータベースを別のターゲット移行先に移行するための概要レポートが作成されます。このレポートを使用して、ターゲットとなる可能性のある移行先を比較し、最適な移行パスを選択できます。

新しいプロジェクトウィザードを実行するには

1. ソースデータベースを選択します。

   a. AWS Schema Conversion Tool を開始します。

   b. [ファイル] メニューで [新しいプロジェクト] を選択します。[データベース移行プロジェクトの新規作成] ダイアログボックスが開きます。

   c. ソースデータベース接続情報を入力するには、以下の手順に従います。

パラメータ	アクション
[Project name] (プロジェクト名)	コンピュータにローカルに保存されているプロジェクトの名前を入力します。
場所	ローカルプロジェクトファイルの場所を入力します。
ソースタイプ	SQL データベース、NoSQL データベース、または ETL のいずれかのオプションを選択します。

パラメータ	アクション
	すべての移行先を含むサマリーレポートを見たい場合は、[SQL Database] を選択してください。
Source engine	ソースデータベースエンジンを選択します。
移行戦略	以下のオプションのいずれかを選択します。 • [エンジンを切り替えてクラウド向けに最適化する] — このオプションはソースデータベースを新しいデータベースエンジンに変換します。 • [同じエンジンを維持しながらクラウド向けに最適化する] — このオプションでは、データベースエンジンをそのまま使用したまま、データベースをオンプレミスからクラウドに移行します。 • [データベースエンジンの切り替えとクラウドの最適化をまとめたレポートを見る] — このオプションでは、利用可能なすべての移行オプションの移行の複雑さを比較しています。 すべての移行先を含む集計評価レポートを見る場合は、最後のオプションを選択してください。

d. [Next] (次へ) をクリックします。[ソースデータベースへの接続] ページが開きます。

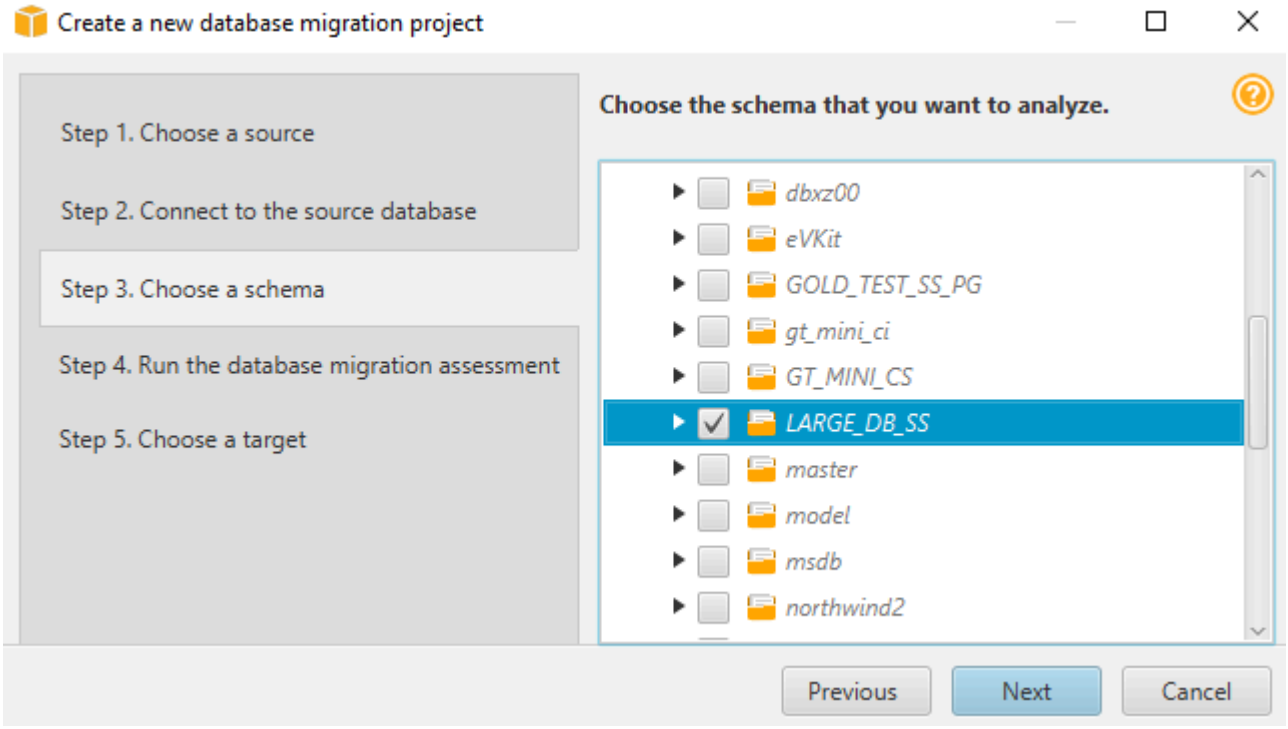
2. ソースデータベースに接続します。

   a. ソースデータベースには接続情報を指定します。接続パラメータはソースデータベースエンジンによって異なります。ソースデータベースの分析に使用するユーザーに、適切な権限があることを確認してください。詳細については、「AWS SCT のソース」を参照してください。

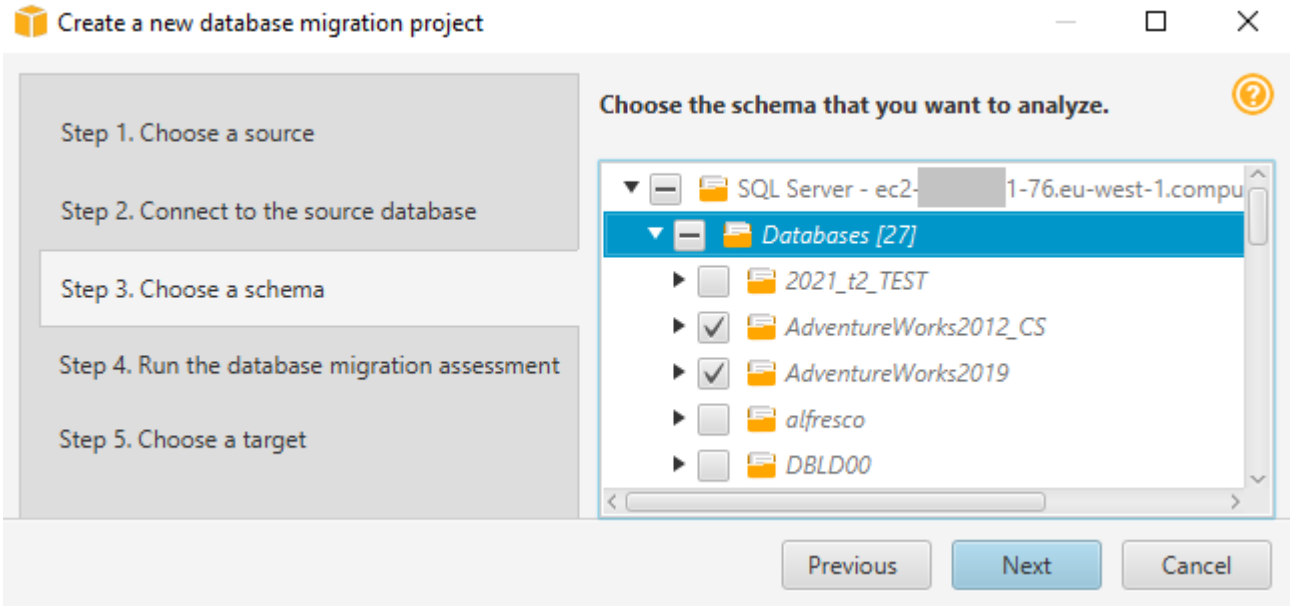
   b. [Next] (次へ) をクリックします。[スキーマの選択] ページが開きます。

3. データベーススキーマを選択します。

   a. 評価するスキーマの名前のチェックボックスをオンにし、スキーマ自体を選択します。選択するとスキーマ名が青く強調表示され、[次へ] ボタンが使用可能になります。



b. 複数のデータベーススキーマを評価する場合は、すべてのスキーマのチェックボックスを選択し、親ノードを選択します。評価を成功させるには、親ノードを選択する必要があります。たとえば、ソース SQL Server データベースの場合は、[データベース] ノードを選択します。親ノードの名前が青く強調表示され、[次へ] ボタンが使用可能になります。



c. [次へ] を選択します。AWS SCT はソースデータベーススキーマを分析し、データベース移行評価レポートを作成します。ソースデータベーススキーマ内のデータベースオブジェクト

の数は、評価の実行にかかる時間に影響します。完了すると、[データベース移行評価の実行] ページが開きます。

4. データベース移行評価を実行します。

   a. さまざまな移行ターゲットの評価レポートを確認して比較したり、評価レポートファイルのローカルコピーを保存してさらに分析したりできます。

   b. データベース移行評価レポートのローカルコピーを保存します。[保存] を選択し、ファイルを保存するフォルダへのパスを入力して [保存] を選択します。AWS SCT は、評価レポートファイルを指定したフォルダに保存します。

   c. [Next] (次へ) をクリックします。[ターゲットの選択] ページが開きます。

5. ターゲットデータベースを選択します。

   a. [ターゲットエンジン] では、評価レポートに基づいて使用することを決定したターゲットデータベースエンジンを選択します。

   b. ターゲットデータベースの接続情報を指定します。表示される接続パラメータは、選択したターゲットデータベースエンジンによって異なります。ターゲットデータベースに指定したユーザーに、必要な権限があることを確認してください。必要な許可の詳細については、AWS SCT のソース および ターゲットとしての Amazon Redshift の許可 のターゲットデータベースの許可について説明しているセクションを参照してください。

   c. [終了] を選択します。AWS SCT はプロジェクトを作成し、マッピングルールを追加します。詳細については、「マッピングルールの作成」を参照してください。

これで、AWS SCT プロジェクトを使用してソースデータベースオブジェクトを変換できます。

AWS SCT プロジェクトの保存とオープン

以下の手順に従って AWS Schema Conversion Tool プロジェクトを保存します。

プロジェクトを保存するには

1. AWS Schema Conversion Tool を開始します。
2. [File] (ファイル) メニューで [Save Project] (プロジェクトの保存) を選択します。

   AWS SCT が、プロジェクトの作成時に指定したフォルダにプロジェクトを保存します。

以下の手順に従って、既存の AWS Schema Conversion Tool プロジェクトを開きます。

プロジェクトを開くには

1. [File] (ファイル) メニューで、[Open project] (プロジェクトを開く) を選択します。[Open] (開く) ダイアログボックスが表示されます。
2. プロジェクトフォルダを選択し、Windows スクリプトコンポーネント (*.sct) ファイルを選択します。
3. AWS SCT はプロジェクトを開きますが、移行元のデータベースと移行先のデータベースには接続しません。データベーススキーマツリーの最上部にある [サーバーに接続] を選択して、ソースデータベースとターゲットデータベースに接続します。

AWS SCT バージョン 1.0.655 以前で保存したプロジェクトを開くと、AWS SCT は、すべてのソースデータベーススキーマのターゲットデータベースプラットフォームに対するマッピングルールを自動的に作成します。他のターゲットデータベースプラットフォームを追加するには、既存のマッピングルールを削除してから、新しいマッピングルールを作成します。マッピングルールの作成に関する詳細については、マッピングルールの作成 を参照してください。

AWS SCT プロジェクトへのデータベースサーバーの追加

複数のソースとターゲットデータベースサーバーを AWS Schema Conversion Tool プロジェクトに追加することができます。

プロジェクトにサーバーを追加するには

1. AWS Schema Conversion Tool を開始します。
2. 新しいプロジェクトを作成するか、既存のプロジェクトを開きます。
3. メニューから [Add source] (ソースの追加) を選択し、新しいソースデータベースを追加します。
4. データベースプラットフォームを選択し、データベース接続の認証情報を指定します。ソースデータベースへの接続に関する詳細については、AWS SCT のソース を参照してください。

以下の手順を使用して、 ソースデータベースに接続します。

データベースに接続するには

1. データベースサーバーのコンテキスト (右クリック) メニューを開き、[Establish connection] (接続を確立する) を選択します。

データベーススキーマツリーの最上部で [サーバーに接続] を選択することもできます。

2. パスワードを入力して、ソースデータベースサーバーに接続します。
3. [接続のテスト] を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
4. [Connect] (接続) を選択して、ソースデータベースに接続します。

次の手順を使用して、AWS SCT プロジェクトからデータベースサーバーを削除します。

データベースサーバーを削除するには

1. 削除するデータベースサーバーを選択します。
2. コンテキスト (右クリック) メニューを開き、[Remove from project] (プロジェクトから削除) を選択します。

   AWS SCT の結果、選択したデータベースサーバー、このサーバーに関連するすべてのマッピングルール、変換結果、およびその他のメタデータが削除されます。

AWS SCT のオフラインモードでの実行

AWS Schema Conversion Tool をオフラインモードで実行できます。以下では、ソースデータベースに接続していないときに既存の AWS SCT プロジェクトを使用する方法について説明します。

次の操作を実行するのに、AWS SCT をソースデータベースに接続する必要はありません。

- マッピングルールを追加します。
- データベース移行評価レポートを作成します。
- データベーススキーマとコードを変換します。
- ソースと変換したコードを編集します。
- ソースと変換したコードを SQL スクリプトとしてテキストファイルに保存します。

AWS SCT をオフラインモードで使用する前に、ソースデータベースに接続し、メタデータを読み込み、プロジェクトを保存してください。このプロジェクトを開くか、ソースデータベースサーバーとの接続を切断して、AWS SCT をオフラインモードで使用します。

AWS SCT をオフラインモードで実行するには

1. AWS Schema Conversion Tool を起動して新しいプロジェクトを作成します。詳細については、「AWS SCT プロジェクトの作成」を参照してください。
2. ソースデータベースサーバーを追加し、ソースデータベースに接続します。詳細については、「AWS SCT プロジェクトへのデータベースサーバーの追加」を参照してください。
3. ターゲットデータベースサーバーを追加するか、仮想ターゲットデータベースプラットフォームを使用します。詳細については、「仮想ターゲットの使用」を参照してください。
4. ソースデータベースのターゲットデータベースプラットフォームを定義するマッピングルールを作成します。詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。
5. [ビュー]、[メインビュー] の順に選択します。
6. ソースデータベースのオブジェクトを表示する左のパネルで、ソースデータベースのスキーマを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[スキーマの変換] を選択します。この操作により、すべてのソーススキーマのメタデータが AWS SCT プロジェクトにロードされます。

   [レポートの作成] 操作と [スキーマの変換] 操作では、すべてのソーススキーマのメタデータも AWS SCT プロジェクトに読み込まれます。コンテキストメニューからこれらの操作のいずれかを実行した場合は、[スキーマのロード] 操作をスキップしてください。
7. [ファイル] メニューで [プロジェクトを保存] を選択し、ソースデータベースのメタデータをプロジェクトに保存します。
8. [サーバーから切断] を選択して、ソースデータベースから切断します。これで、AWS SCT をオフラインモードで使用できるようになりました。

AWS SCT ツリーフィルターの使用

AWS SCT は、ソースからターゲットにデータを移行する場合、すべてのメタデータをソースおよびターゲットデータベースからツリー構造内にロードします。この構造は、AWS SCT のメインプロジェクトウィンドウにツリービューとして表示されます。

一部のデータベースは、ツリー構造内のオブジェクト数が非常に多くなる場合があります。 のツリーフィルタAWS SCTを使用すると、ソースおよびターゲットツリー構造内のオブジェクトを検索できます。ツリーフィルタでは、オブジェクトは変更されません。データベースの変換に伴ってオブジェクトが変換される場合とは異なります。フィルターは、ツリー内の表示のみを変更します。

ツリーフィルターは、AWS SCT でロード済みのオブジェクトを対象とします。つまり、検索中は AWS SCT でデータベースからオブジェクトがロードされません。そのため、通常、ツリー構造内のオブジェクト数はデータベース内のオブジェクト数より少なくなります。

ツリーフィルタについては、以下の点に注意してください。

- フィルタのデフォルトは ANY です。このフィルタでは、名前の検索を使用してオブジェクトを見つけます。
- オブジェクトタイプを選択すると、選択したタイプのオブジェクトのみが表示されます。
- フィルタマスクを使用すると、さまざまなタイプの記号 (Unicode、スペース、特殊文字など) を表示できます。「%」文字は、あらゆる記号のワイルドカードです。
- フィルタを適用すると、フィルタしたオブジェクトの数のみがカウントに表示されます。

ツリーフィルタを作成するには

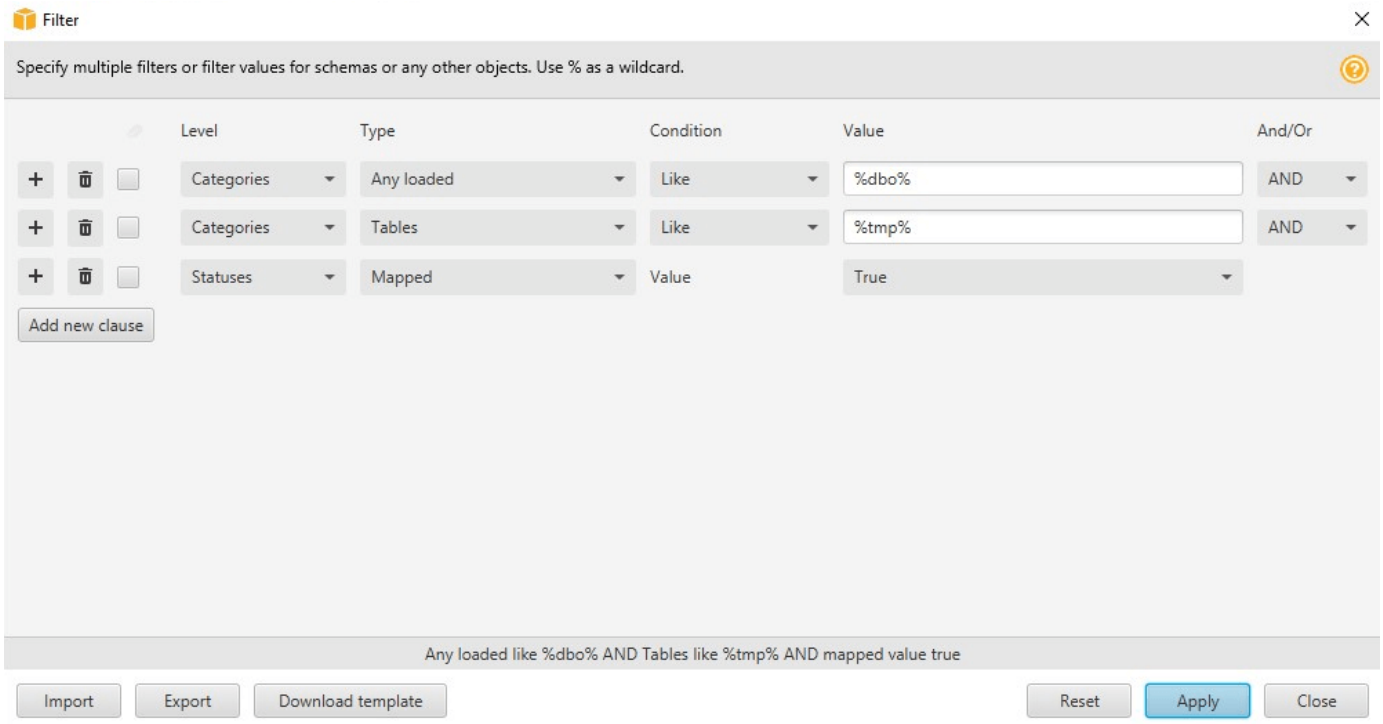
1. 既存の AWS SCT プロジェクトを開きます。
2. ツリーフィルターを適用するデータベースに接続します。
3. フィルタアイコンを選択します。

   フィルタを元に戻すアイコンは、現在適用されているフィルタがないため、灰色で表示されます。
4. [Filter] (フィルター) ダイアログボックスに、以下の情報を入力します。ダイアログボックスのオプションは、データベースエンジンごとに異なります。

AWS SCT フィルターオプション	アクション
[レベル]	[Categories] (カテゴリー) を選択して、オブジェクトをカテゴリ別にフィルター処理します。 [Statuses] (ステータス) を選択して、オブジェクトをステータス別にフィルター処理します。
タイプ	[Level] (レベル) の[Categories] (カテゴリー) の場合、フィルター処理されたオブジェクトのカテゴリーを選択します。[Any loaded] (任意

AWS SCT フィルターオプション	アクション
	のロード済み) をクリックして、すべてのカテゴリーのオブジェクトを表示します。 [Level] (レベル) の[Statuses] (ステータス) の場合、フィルター処理されたオブジェクトのステータスを選択します。次のオプションのいずれかを選択します。 • [Converted] (変換済み) で、変換されたすべてのオブジェクトを表示 • [Has actions] (アクションあり) で、変換上の問題があるすべてのオブジェクトを表示 • [Encrypted] (暗号済み) で暗号化されたすべてのオブジェクトを表示
条件	[Level] (レベル) の [Categories] (カテゴリー) の場合、[Like] (次のパターンと一致する) および [Not like] (次のパターンと一致しない) のいずれかのフィルター条件を選択します。 [Level] (レベル) の [Statuses] (ステータス) の場合、フィルター条件オプションは使用できません。
値	[Level] (レベル) の [Categories] (カテゴリー) の場合、ツリーをフィルター処理する[Value] (値) を入力します。 パーセント (%) をワイルドカードとして使用すると、すべてのオブジェクトが表示されます。 [Level] (レベル) の [Statuses] (ステータス) の場合、True または False の [Value] (値) を選択します。
And/Or	AND または OR の論理演算子を選択して、複数のフィルター句を適用します。



5. [Add new clause] (新しい句を追加) を選択して、さらにフィルター句を追加します。AWS SCT では、複数の AND または OR 論理演算子を使用して、複数のフィルター句を適用できます。
6. [Apply] (適用) を選択します。[Apply] (適用) を選択すると、フィルタを元に戻すアイコン (フィルタアイコンの横) が有効になります。適用したフィルタを削除する場合は、このアイコンを使用します。
7. [Close] (閉じる) を選択してダイアログボックスを閉じます。

ツリーに表示されるスキーマをフィルタリングする場合、スキーマを変換するときに変換されるオブジェクトは変更されません。フィルタにより、ツリー内の表示が変更されるのみです。

ツリーフィルターのファイルリストのインポート

セミコロン区切りを含むカンマ区切り値 (CSV) ファイル、またはツリー フィルターで使用する名前または値を含む JSON ファイルをインポートできます。既存の AWS SCT プロジェクトを開き、ツリーフィルターを適用するデータベースに接続して、フィルターアイコンを選択します。

ファイル例をダウンロードするには、[Download template] (テンプレートのダウンロード) を選択します。ファイルの名前を入力し、[保存] を選択します。

既存のフィルター設定をダウンロードするには、[Export] (エクスポート) を選択します。ファイルの名前を入力し、[保存] を選択します。

ツリーフィルターのファイルリストをインポートするには、[Import] (インポート) を選択します。インポートするファイルを選択し、[Open] (開く) を選択します。[Apply] (適用)、[Close] (終了) の順に選択します。

CSV ファイルでは区切り文字としてセミコロンが使用され、次の形式になっています。

- `object_type` は、検索するオブジェクトのタイプです。
- `database_name` は、このオブジェクトがあるデータベースの名前です。
- `schema_name` は、このオブジェクトがあるスキーマの名前です。
- `object_name` は、オブジェクト名です。
- `import_type` は、この項目をフィルターから `include` または `exclude` するかどうかを指定します。

JSON ファイルを使用して、ネスト化されたルールなどの複雑なフィルタリングケースを記述します。JSON ファイルの形式は次の通りです。

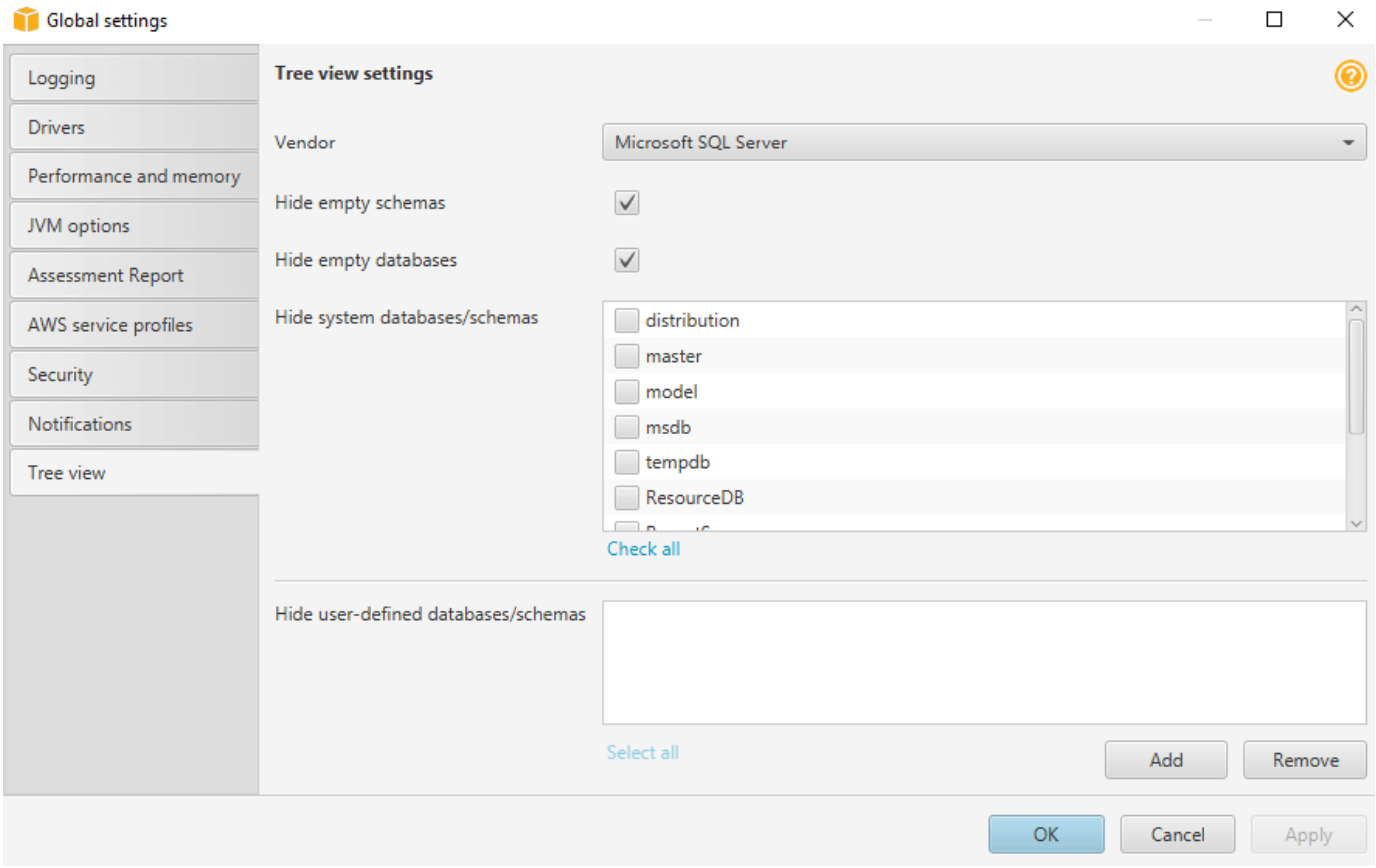
- `filterGroupType` は、複数のフィルタ句に適用されるフィルタールールのタイプ (`AND` または `OR` 論理演算子)です。
- `filterCategory` は、フィルターのレベルです ([Categories] (カテゴリー) または [Statuses] (ステータス))。
- `names` は、[Categories] (カテゴリー) フィルターに適用されるオブジェクト名のリストです。
- `filterCondition` は [Categories] (カテゴリー) フィルターに適用されるフィルター条件 (`LIKE` または `NOT LIKE`) です。
- `transformName` は、[Status] (ステータス) フィルターに適用されるステータス名です。
- `value` は、ツリーをフィルター処理する値です。
- `transformValue` は [Status] (ステータス) フィルターに適用されるフィルターの値 (`TRUE` または `FALSE`) です。

AWS SCT ツリービューのスキーマの非表示

ツリービューの設定では、AWS SCT ツリービューに表示するスキーマとデータベースを指定します。空のスキーマ、空のデータベース、システムデータベース、ユーザー定義のデータベースやスキーマは、非表示にすることができます。

ツリービューのデータベースやスキーマを非表示にするには

1. AWS SCT プロジェクトを開きます。
2. ツリービューで表示するデータストアに接続します。
3. [Settings] (設定)、[Global Settings] (グローバル設定)、[Tree View] (ツリービュー) の順に選択します。



4. [Tree View Settings] (ツリービューの設定) セクションで、以下の操作を行います。
   - [Vendor] (ベンダー) で、データベースプラットフォームを選択します。
   - [Hide empty schemas] (空のスキーマを非表示にする) をクリックして、選択したデータベースプラットフォームの空のスキーマを非表示にします。

- [Hide empty databases] (空のデータベースを非表示にする) をクリックして、選択したデータベースプラットフォームの空のデータベースを非表示にします。
- [Hide System Databases/Schemas] (システムデータベース/スキーマを非表示にする) で、非表示にするシステムデータベースやスキーマの名前を選択します。
- [Hide User Defined Databases/Schemas] (ユーザー定義のデータベース/スキーマを非表示にする) で、非表示にするユーザー定義のスキーマやデータベースの名前を入力し、[Add] (追加) を選択します。名前の大文字と小文字は区別されません。

5. [OK] をクリックします。

データベース移行評価レポートの作成と確認

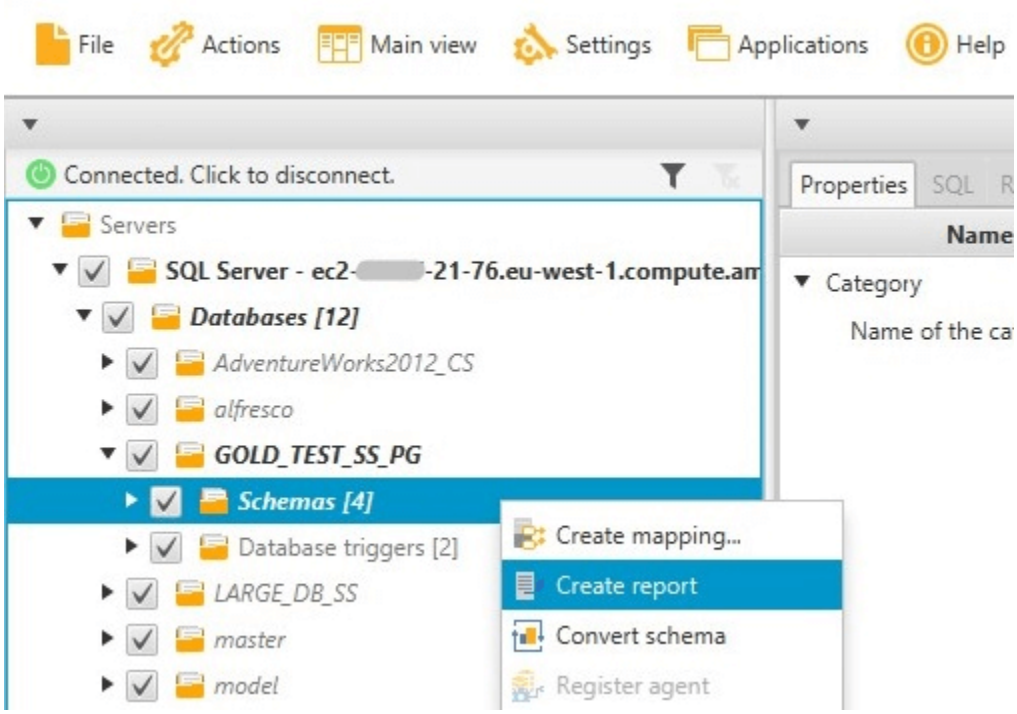
データベース移行評価レポートには、ターゲットとなる Amazon RDS DB インスタンスのエンジン用に自動変換できないスキーマに対するすべてのアクション項目が、要約されて表示されます。また、ターゲット DB インスタンス用の同等のコードを記述するのにかかる推定労力も示されます。

ソースデータベースとターゲットプラットフォームをプロジェクトに追加し、マッピングルールを指定した後、データベース移行評価レポートを作成できます。

データベース移行評価レポートを作成および表示するには

1. 評価レポートを作成するソースデータベーススキーマのマッピングルールを作成したことを確認してください。詳細については、「新しいマッピングルールの追加」を参照してください。
2. [View] (ビュー) メニューで、[Main View] (メインビュー) を選択します。
3. ソースデータベースのスキーマを表示する左のパネルで、評価レポートを作成するスキーマオブジェクトを選択します。

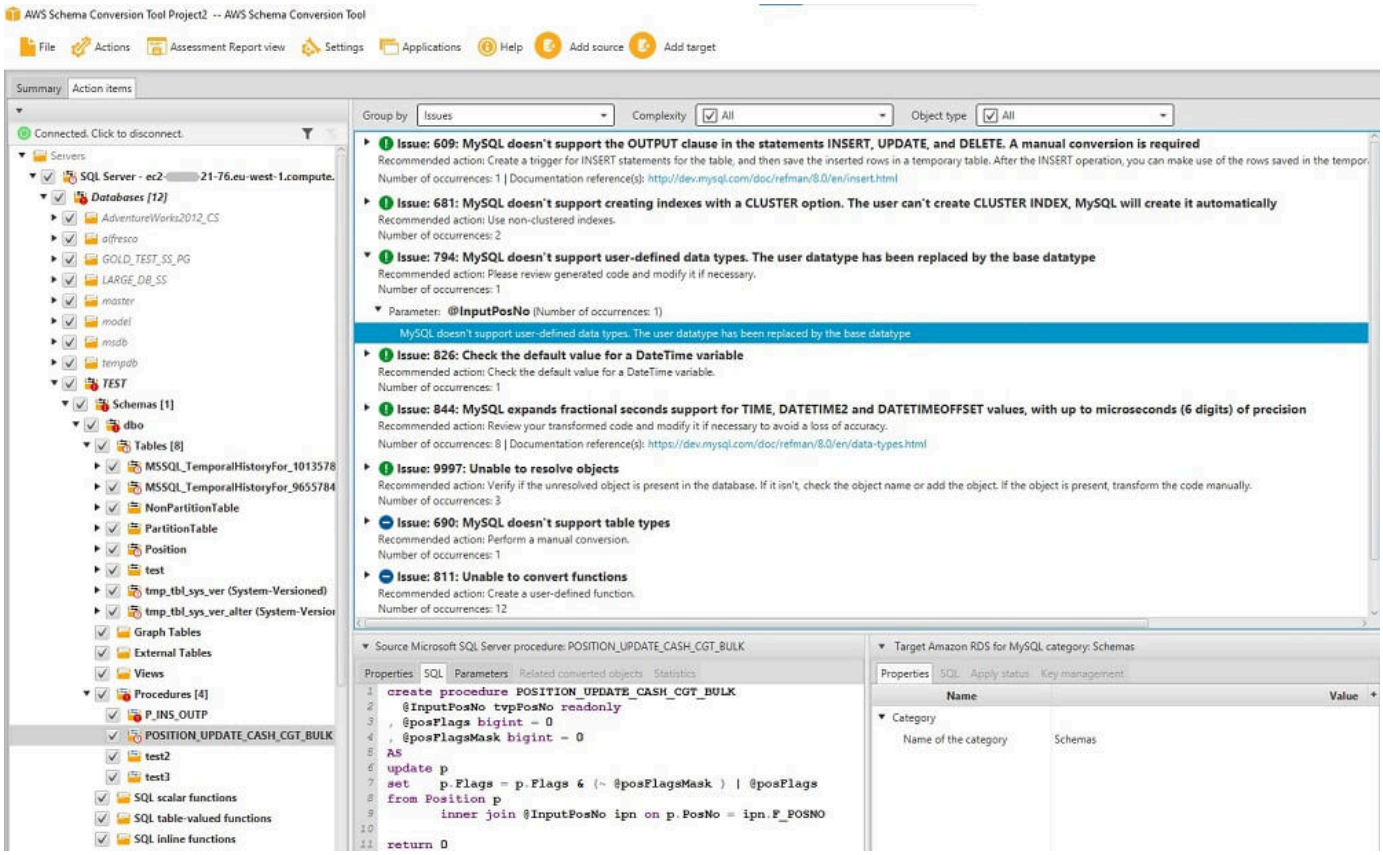
   評価レポートを作成するすべてのスキーマオブジェクトのチェックボックスがオンになっていることを確認してください。
4. オブジェクトのコンテキスト (右クリック) メニューを開き、[レポートの作成] を選択します。



評価レポートビューが開きます。

5. [アクション項目] タブを選択します。

   [アクション項目] タブには、自動変換できないスキーマについて説明する項目のリストが表示されます。リストからいずれかのアクション項目を選択します。アクション項目が適用されるスキーマが AWS SCT によってハイライトされます。

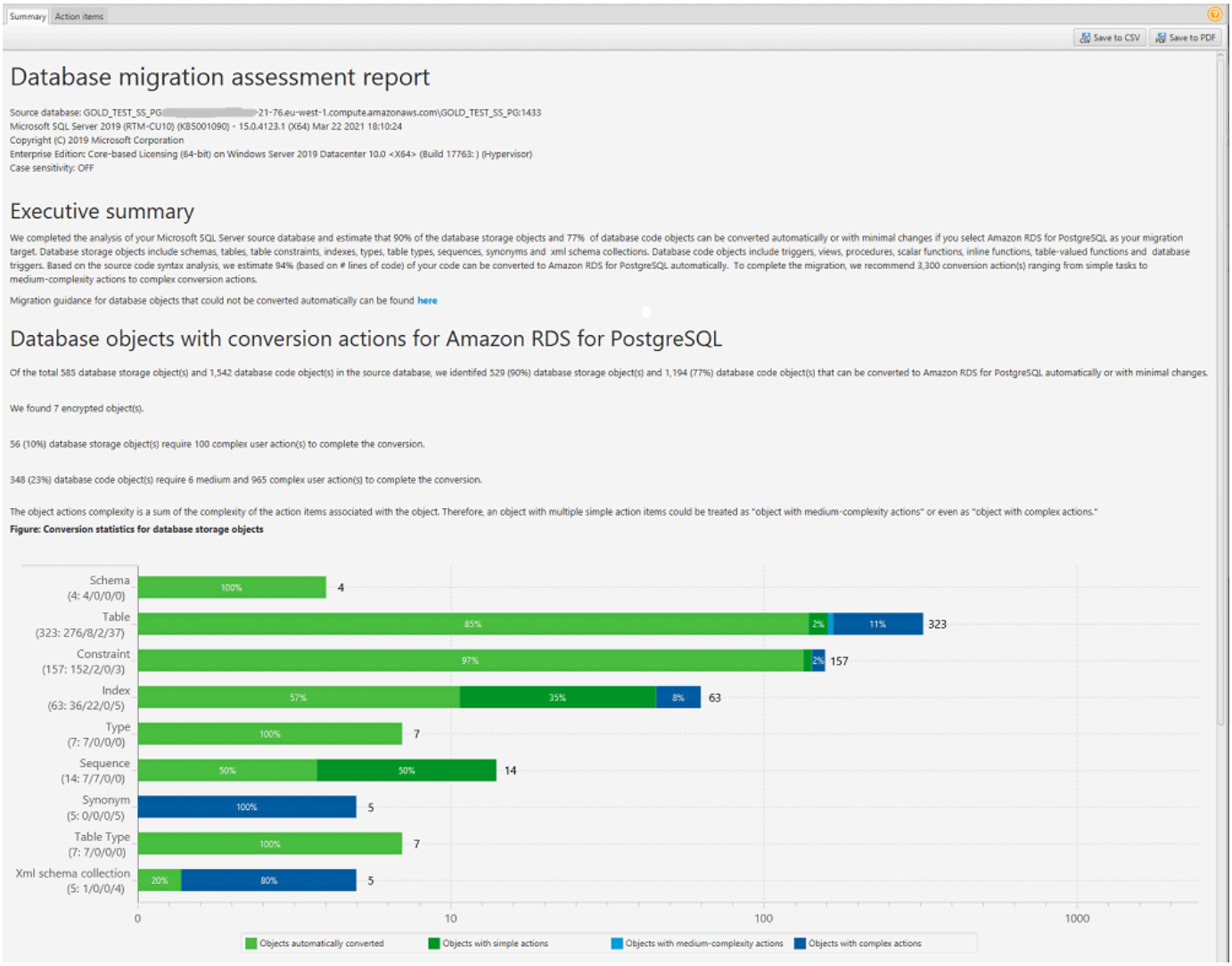


6. [Summary] (概要) タブを選択します。

   [Summary] (概要) タブには、データベース移行評価レポートの要約が表示されます。そのタブには、自動変換された項目の数と自動変換されなかった項目の数が示されます。要約には、ターゲット DB インスタンスで、ソースデータベースのものと同等のスキーマを作成するのにかかる推定時間も示されます。

   [License Evaluation and Cloud Support] (ライセンス評価およびクラウドサポート) セクションには、同エンジンを実行する Amazon RDS DB インスタンスへの既存のオンプレミスのデータベーススキーマの移動に関する情報が含まれます。例えば、ライセンスタイプを変更する場合は、このレポートセクションを確認することで、現在のデータベースから削除する必要のある機能を見極めることができます。

   評価レポートの要約のサンプルを以下に表示します。



7. [Summary] (概要) タブ、[Save to PDF] (PDF に保存) の順に選択します。データベース移行評価レポートは PDF ファイルとして保存されます。PDF ファイルには、要約とアクション項目の情報の両方が含まれます。

   [CSV に保存] を選択して、レポートをカンマ区切り値 (CSV) ファイルとして保存することもできます。このオプションを選択すると、AWS SCT で 3 つの CSV ファイルが作成されます。これらのファイルには、次の情報が含まれています。

   - 推奨されるアクションが含まれる変換アクション項目のリスト。
   - 変換アクション項目の概要と、そのアクション項目の発生を変換するために必要となる労力の見積もり。
   - 変換にかかる予測推定時間別に分類された多数のアクション項目が含まれるエグゼクティブサマリー。

Database objects with conversion actions for Amazon RDS for PostgreSQL

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identifed 529 (90%) database storage object(s) and 1,194 (77%) database code object(s) that can be converted to Amazon RDS for PostgreSQL automatically or with minimal changes.

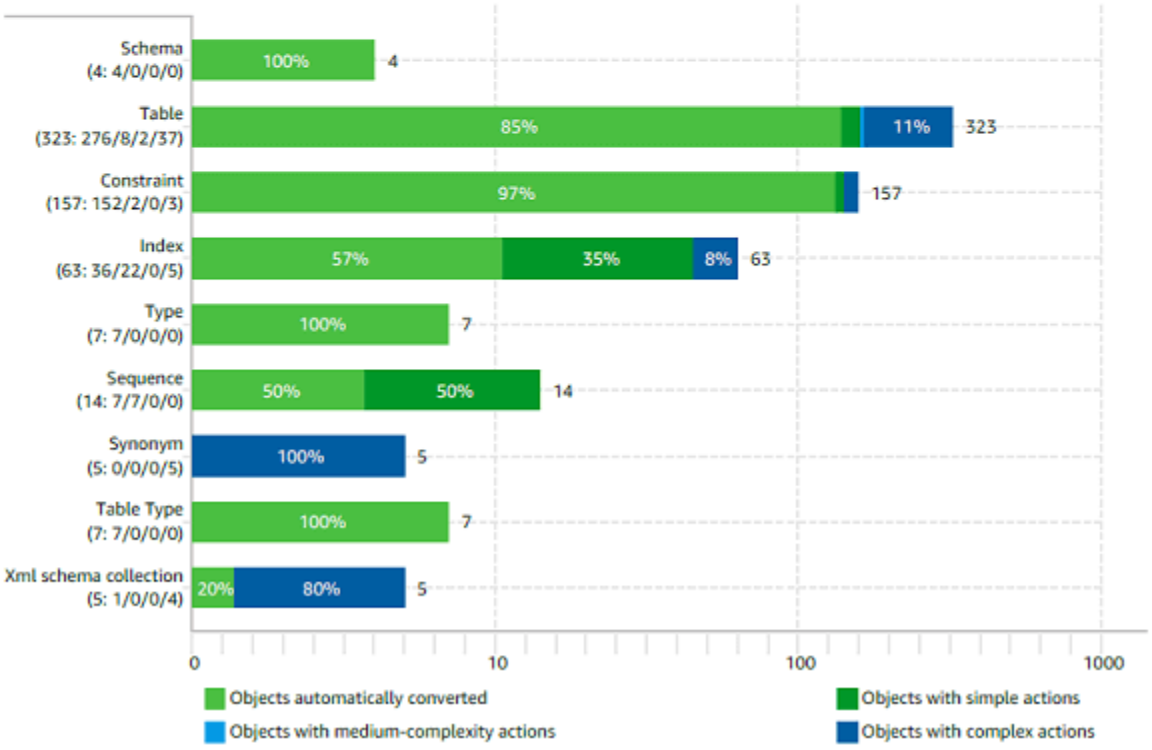
We found 7 encrypted object(s).

56 (10%) database storage object(s) require 100 complex user action(s) to complete the conversion.

348 (23%) database code object(s) require 6 medium and 965 complex user action(s) to complete the conversion.

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Figure: Conversion statistics for database storage objects

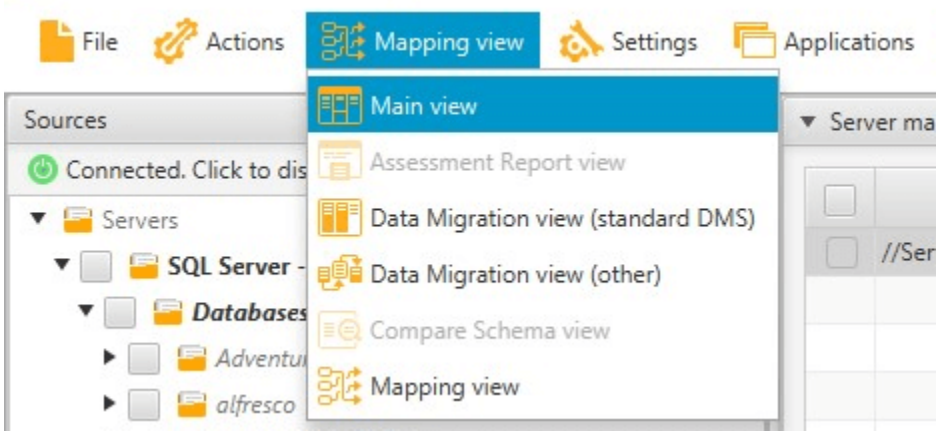


スキーマの変換

ソースとターゲットのデータベースをプロジェクトに追加し、マッピングルールを作成した後、ソースデータベーススキーマを変換できます。以下の手順を使用して、スキーマを変換します。

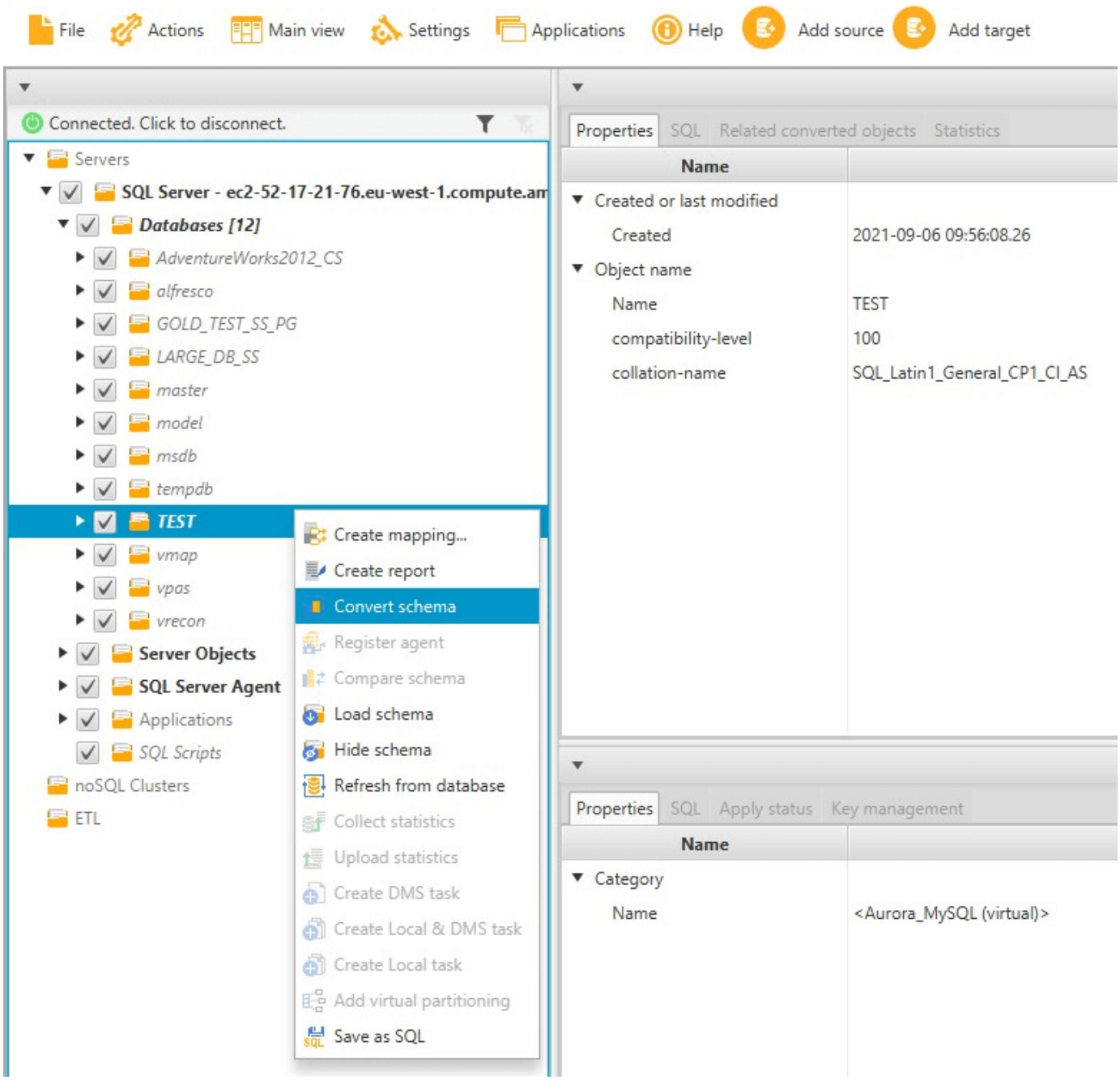
スキーマを変換するには

1. [ビュー]、[メインビュー] の順に選択します。



2. ソースデータベースのスキーマが表示される左側のパネルで、変換するオブジェクトの名前のチェックボックスをオンにします。次に、このオブジェクトを選択します。AWS SCT はオブジェクト名を青で強調表示します。オブジェクトのコンテキスト (右クリック) メニューを開き、[スキーマの変換] を選択します。

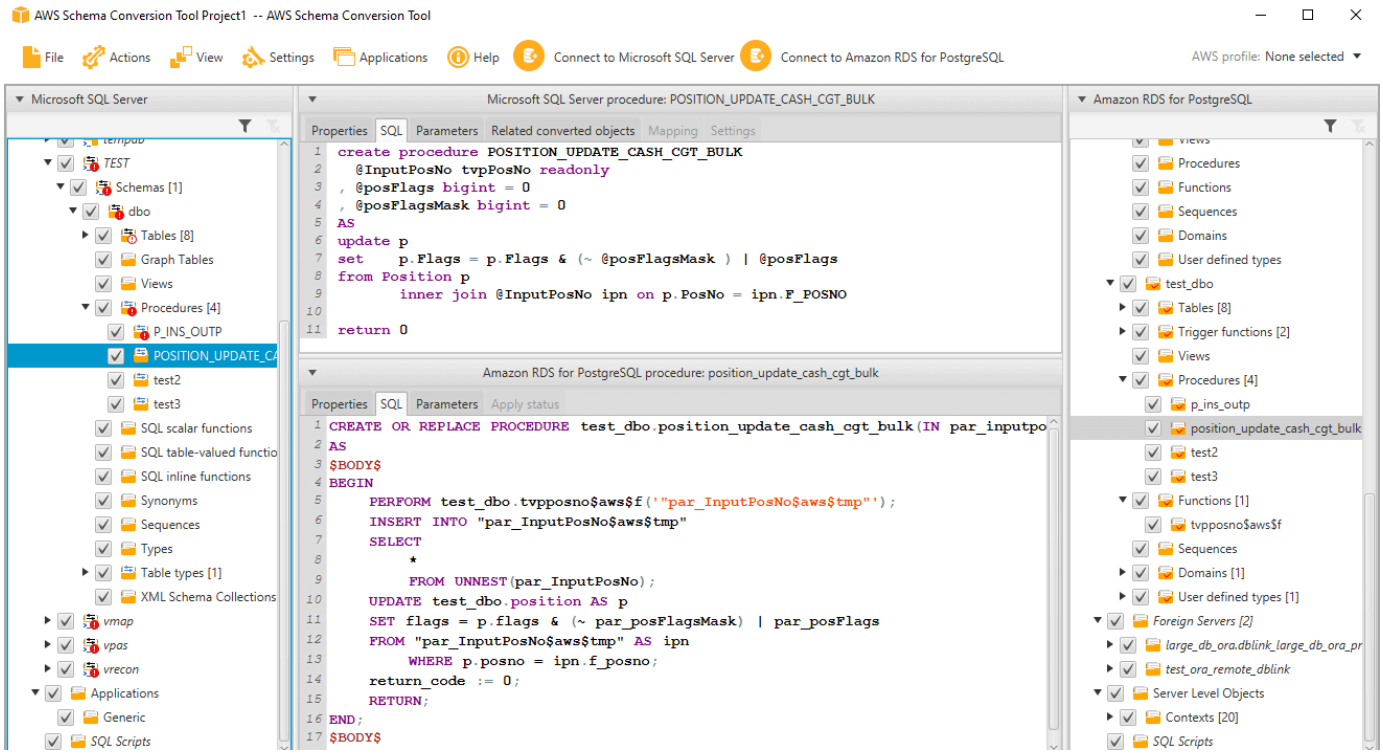
   複数のデータベースオブジェクトを変換するには、すべてのオブジェクトのチェックボックスを選択します。次に、親ノードを選択します。たとえば、テーブルの場合、親ノードは [テーブル] です。AWS SCT で親ノードの名前が青色で強調表示されていることを確認してください。オブジェクトのコンテキスト (右クリック) メニューを開き、[スキーマの変換] を選択します。



3. AWS SCT でスキーマの変換が終了すると、プロジェクトの右のパネルでスキーマの候補が表示されます。

   この時点で、スキーマはターゲットのデータベースインスタンスには適用されていません。計画したスキーマはプロジェクトの一部です。変換されたスキーマ項目を選択すると、ターゲットのデータベースインスタンスの下中央のパネルに、計画したスキーマコマンドが表示されます。

このウィンドウでスキーマを編集できます。変換されたスキーマの適用を選択すると、編集したスキーマはプロジェクトの一部として保存され、ターゲットのデータベースインスタンスに書き込まれます。



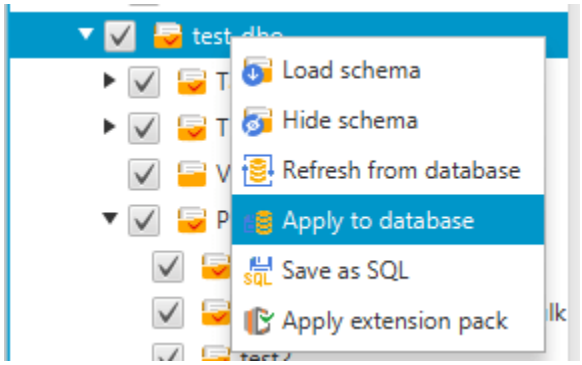
ターゲット DB インスタンスへの変換されたスキーマの適用

変換されたデータベーススキーマをターゲット DB インスタンスに適用できます。スキーマがターゲット DB インスタンスに適用されたら、データベース移行評価レポートのアクション項目に基づいてスキーマを更新できます。

> **Warning**
>
> この手順により、既存のターゲットスキーマは上書きされます。意図せずスキーマを上書きしないように注意してください。ターゲット DB インスタンスのスキーマに変更を加えている場合、その変更も上書きされます。

変換されたデータベーススキーマをターゲットのデータベースインスタンスに適用するには

1. プロジェクトの右側のパネルの上部にある [サーバーに接続] を選択して、ターゲットデータベースに接続します。ターゲットデータベースに接続している場合は、この手順をスキップしてください。
2. ターゲット DB インスタンスに計画したスキーマを表示するスキーマ要素を、プロジェクトの右パネルで選択します。
3. スキーマ要素のコンテキスト (右クリック) メニューを開き、[Apply to database] (データベースに適用) を選択します。



変換されたスキーマがターゲット DB インスタンスに適用されます。

AWS サービスプロファイルの AWS SCT への保存

AWS SCT に AWS の認証情報を保存できます。AWS SCT では、AWS のサービスと統合されている機能を使用するときに、認証情報を使用します。たとえば、AWS SCT は Amazon S3、AWS Lambda、Amazon Relational Database Service (Amazon RDS)、AWS Database Migration Service (AWS DMS) と統合します。

認証情報が必要な機能にアクセスすると、AWS SCT より AWS 認証情報を求められます。認証情報はグローバルアプリケーション設定で保存できます。保存された認証情報は、AWS SCT の認証情報入力画面で選択できます。

グローバルアプリケーション設定では、別の組み合わせの AWS 認証情報を保存することもできます。例えば、テストシナリオで使用する認証情報とは別に、運用シナリオで使用する認証情報を保存できます。また、AWS リージョン ごとに異なる認証情報を保存することもできます。

AWS 認証情報の保存

AWS 認証情報をグローバルに保存するには、以下の手順を行います。

AWS 認証情報を保存するには

1. AWS Schema Conversion Tool を開始します。
2. [設定] メニューを開き、[グローバル設定] を選択します。[Global settings] (グローバル設定) ダイアログボックスが表示されます。
3. [AWSサービスプロファイル] を選択し、[AWS新しいサービスプロファイルの追加] を選択します。
4. 次のように AWS 情報を入力します。

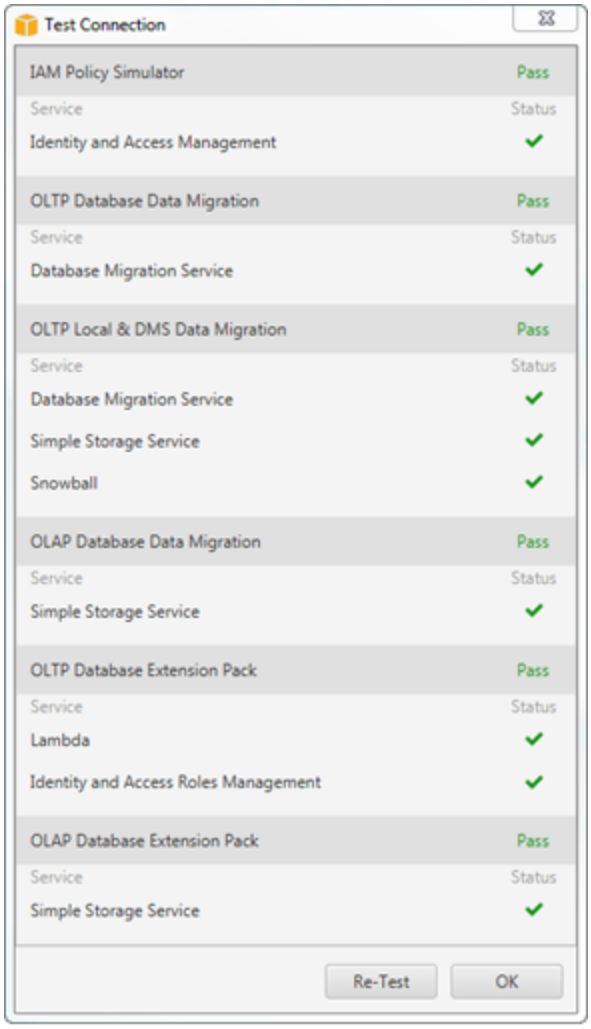
AWS SCT オプション	アクション
プロファイル名	プロファイルの名前を入力します。
[AWS アクセスキー]	AWS アクセスキーを入力します。
AWS シークレットキー	AWS シークレットアクセスキーを入力します。AWS アクセスキーの詳細については、『IAM ユーザーガイド』の「IAM ユーザー」の「アクセスキーの管理」を参照してください。
地域	AWS リージョン でプロファイルを選択します。
[Amazon S3 バケットフォルダ]	プロファイルで [Amazon S3 bucket] (Amazon S3 バケット) を選択します。Amazon S3 に接続する機能を使用する場合に限り、バケットを指定する必要があります。必要な権限の詳細については、「AWS サービスプロファイルを使用するための権限」を参照してください。

連邦情報処理標準 (FIPS) のセキュリティ要件に準拠する場合は、[Use FIPS endpoint for S3] (S3に FIPS エンドポイントを使用) を選択します。FIPS エンドポイントは、以下の AWS リージョンで利用できます。

- US East (N. Virginia) Region
- US East (Ohio) Region
- 米国西部 (北カリフォルニア) リージョン
- 米国西部 (オレゴン) リージョン

5. [テスト接続] を選択して、認証情報が正しくアクティブであることを確認します。

   [テスト接続] ダイアログボックスが表示されます。プロファイルに接続されたサービスのそれぞれについて、ステータスを確認できます。[Pass] (パス) は、プロファイルが正常にサービスにアクセスできることを示します。



6. プロファイルを設定したら、[Save] (保存) を選択してプロファイルを保存します。変更をキャンセルする場合は、[Cancel] (キャンセル) を選択します。
7. [OK] を選択し、[グローバル設定] ダイアログボックスを閉じます。

プロジェクトのデフォルトプロファイルの設定

AWS SCT プロジェクトのデフォルトプロファイルを設定できます。この関連付けを行うと、プロファイルに保存されている AWS 認証情報がプロジェクトに関連付けられます。プロジェクトが開いたら、次の手順でデフォルトプロファイルを設定します。

プロジェクトのデフォルトプロファイルを設定するには

1. AWS Schema Conversion Tool を起動して新しいプロジェクトを作成します。
2. [Settings] (設定) メニューから [Project settings] (プロジェクト設定) を選択します。[Project settings] (プロジェクト設定) ダイアログボックスが表示されます。
3. [プロジェクト環境] タブを選択します。
4. [新しいAWS サービスプロファイルの追加] をクリックして、新しいプロファイルを追加します。[AWS サービスプロファイル] で、プロジェクトと関連付けるプロファイルを選択します。
5. [OK] を選択し、[Project settings] (プロジェクト設定) ダイアログボックスを閉じます。変更をキャンセルする場合は、[Cancel] (キャンセル) を選択します。

AWS サービスプロファイルを使用するための権限

AWS サービスプロファイルから Amazon S3 バケットにアクセスするには、以下の権限が必要です。

- `s3:PutObject` — Amazon S3 バケットにオブジェクトを追加します。
- `s3:DeleteObject` — オブジェクトの null バージョンを削除し、削除マーカーを挿入します。これがオブジェクトの現在のバージョンになります。
- `s3:ListBucket` — Amazon S3 バケットから最大 1,000 個のオブジェクトを返します。
- `s3:GetObject` — Amazon S3 バケットからオブジェクトを取得できます。

次に、ユーザーにアクセス許可を付与するコード例を示します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "*"
            ]
```

```
        }
    ]
}
```

AWS Secrets Manager を使用する

AWS SCT は、ユーザーが AWS Secrets Manager に保存したデータベース認証情報を使用できます。Secrets Manager のデータベース接続ダイアログボックスにすべての値を入力できます。Secrets Manager を使用するには、必ず AWS プロファイルを AWS Schema Conversion Tool に保存してください。

AWS Secrets Manager の詳細については、『AWS Secrets Manager ユーザーガイド』の「AWS Secrets Manager とは」を参照してください。AWS プロファイルの詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。

Secrets Manager からデータベース認証情報を取得するには

1. AWS Schema Conversion Tool を起動して新しいプロジェクトを作成します。
2. [ソースの追加] または [ターゲットの追加] を選択して、新しいソースデータベースを プロジェクトに追加します。
3. データベースプラットフォームを選択し、[次へ] を選択します。
4. [AWS シークレット] で、使用するシークレットを選択します。
5. [入力] を選択します。次に、AWS SCT はデータベース接続ダイアログボックスにすべての値を入力します。
6. [接続のテスト] を選択して、AWS SCT がデータベースに正常に接続できることを確認します。
7. [接続] を選択して、データベースに接続します。

AWS SCT は、次の構造を持つシークレットをサポートします。

```
{
  "username": "secret_user",
  "password": "secret_password",
  "engine": "oracle",
  "host": "secret_host.eu-west-1.compute.amazonaws.com",
  "port": "1521",
  "dbname": "ora_db"
}
```

この構造では、username と password の値は必須で、その他の値はすべてオプションです。Secrets Manager に保存する値には、すべてのデータベース認証情報が含まれていることを確認してください。

データベースパスワードの保存

データベースパスワードまたは SSL 証明書を AWS SCT キャッシュに保存できます。パスワードを保存するには、接続の作成時に [Store Password] (パスワードの保存) を選択します。

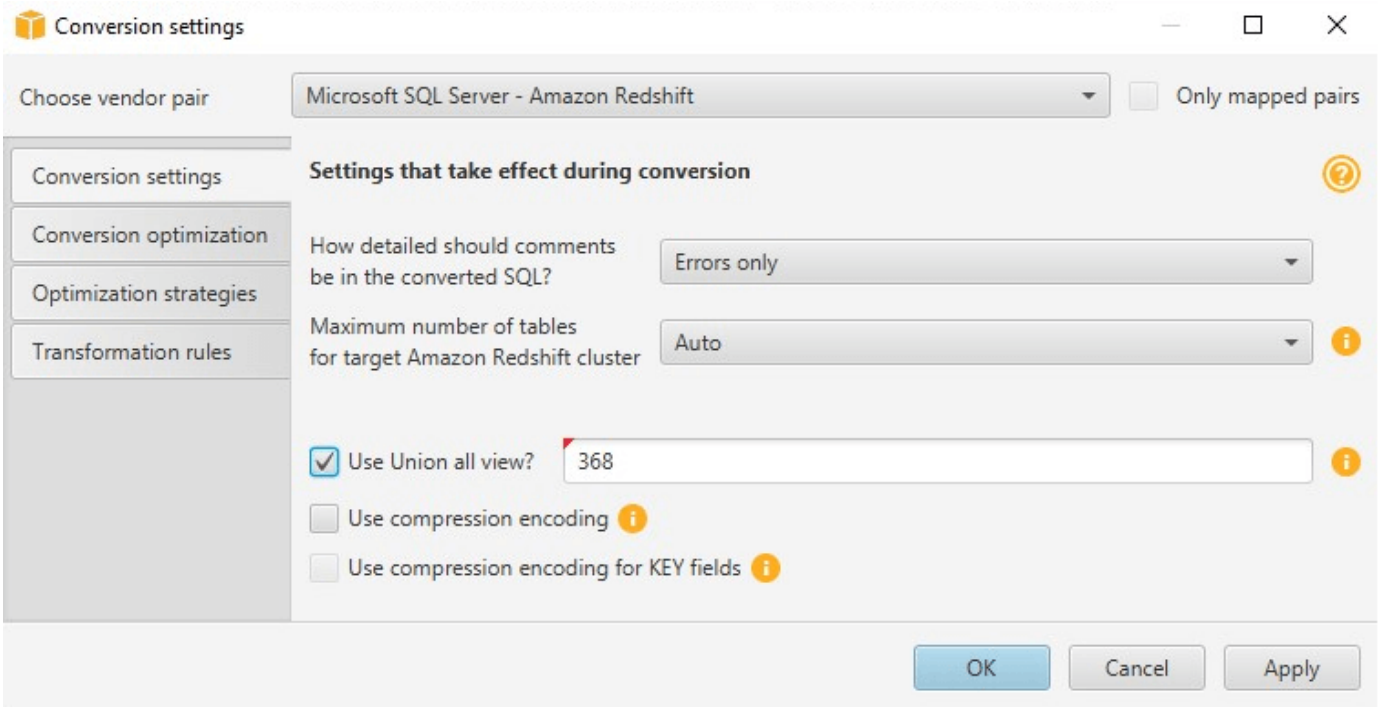
パスワードは、ランダムに生成されたトークンを使用して seed.dat ファイルに暗号化されます。次に、パスワードはユーザー名と共にキャッシュファイルに保存されます。seed.dat ファイルを失うか、破損した場合、データベースパスワードの暗号化が正しく解除されない可能性があります。この場合、接続は失敗します。

パーティション分割されたテーブルでのプロジェクトの Union All ビューの使用

ソーステーブルがパーティション分割されている場合、AWS SCT はn 個のターゲットテーブルを作成します (但し、n はソーステーブルのパーティション数)。AWS SCT は、ターゲットテーブルの上に UNION ALL ビューを作成して、ソーステーブルを表示します。AWS SCT データ抽出プログラムを使用してデータを移行すると、ソーステーブルのパーティションが抽出され、個々のサブタスクによって並列にロードされます。

プロジェクトで Union All ビューを使用するには

1. AWS SCT を起動します。新しいプロジェクトを作成するか、既存の AWS SCT プロジェクトを開きます。
2. [Settings] (設定) メニューを開き、[Global Settings] (グローバル設定) を選択します。
3. 上部にあるリストから、OLAP データベースのペアを選択します。
4. [Union all ビューを使用しますか？] をオンにします。



5. [OK] を選択して設定を保存し、[Current project settings] (現在のプロジェクト設定) ダイアログボックスを閉じます。

AWS SCT のキーボードショートカット

AWS SCT で使用できるキーボードショートカットは次のとおりです。

キーボードショートカット	説明
Ctrl+N	新しいプロジェクトを作成します。
Ctrl+O	既存のプロジェクトを開きます。
Ctrl+S	開いているプロジェクトを保存します。
Ctrl+W	ウィザードを使用して新しいプロジェクトを作成します。
Ctrl+M	新しいマルチサーバー評価を作成します。
Ctrl+L	新しいソースデータベースを追加します。
Ctrl+R	新しいターゲットデータベースを追加します。

キーボードショートカット	説明
Ctrl+F4	開いているプロジェクトを閉じます。
F1	AWS SCT ユーザーガイドを開きます。

AWS SCT の開始方法

ソースデータベースのスキーマを変換するには、AWS Schema Conversion Tool (AWS SCT) を使用できます。ソース データベースは、オンプレミスまたは Amazon EC2 インスタンス上で実行される自己管理型エンジンにすることができます。ソーススキーマは、AWS でホストされるサポートされている任意のデータベースのスキーマに変換できます。AWS SCT アプリケーションは、プロジェクトベースのユーザーインターフェイスを提供します。

AWS SCT で行う大半の作業は、以下のステップで開始します。

1. AWS SCT をインストールします。詳細については、「のインストール、検証、更新 AWS SCT」を参照してください。
2. 必要に応じて AWS SCT エージェントをインストールします。AWS SCT エージェントは、異種混在型のソースとターゲットの間など、特定の移行シナリオにのみ必要です。詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。
3. AWS SCT のユーザーインターフェイスについて理解します。詳細については、「AWS SCT ユーザーインターフェイスの使用」を参照してください。
4. AWS SCT プロジェクトを作成します。ソースおよびターゲットデータベースに接続します。ソースデータベースへの接続の詳細については、「AWS SCT のソース」を参照してください。
5. マッピングルールを作成します。テーブルマッピングの詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。
6. データベース移行評価レポートを実行し、確認します。評価レポートの詳細については、「データベース移行評価レポートの作成と確認」を参照してください。
7. ソースデータベーススキーマを変換します。変換には、変換されない項目を処理する方法や、特定の方法で変換する必要がある項目をマッピングする方法など、留意する必要がある側面がいくつかあります。ソーススキーマの変換の詳細については、「AWS SCT を使用したデータベーススキーマの変換」を参照してください。

   データウェアハウススキーマを変換する場合も、変換を行う前に考慮する必要がある側面があります。詳細については、「AWS SCT を使用したデータウェアハウススキーマの Amazon Redshift への変換」を参照してください。
8. スキーマの変換をターゲットに適用します。ソーススキーマ変換の適用の詳細については、「変換したコードの適用」を参照してください。

9. AWS SCT は、SQL ストアドプロシージャおよびその他のアプリケーションコードを変換するためにも使用できます。詳細については、「AWS SCT を使用したアプリケーション SQL の変換」を参照してください。

AWS SCT を使用して、ソースデータベースから Amazon が管理するデータベースにデータを移行することもできます。例については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。

AWS SCT のソース

AWS Schema Conversion Tool (AWS SCT) は、次のソースデータベースおよびデータウェアハウスのスキーマをターゲットデータベースまたはデータウェアハウスに変換できます。アクセス権限、接続、および AWS SCT がターゲットデータベースまたはデータウェアハウスで使用するために変換できるものについては、次のトピックの詳細を参照してください。

暗号化情報

Amazon RDS 接続の暗号化

データベースソース

- Apache Cassandra をソースとしての使用
- Azure SQL データベースのソースとしての使用
- IBM Db2 for z/OS をソースとして使用する
- IBM Db2 LUW をソースとして使用する
- ソースとしての MySQL の使用
- ソースとしての Oracle Database の使用
- PostgreSQL のソースとしての使用
- ソースとしての SAP ASE (Sybase ASE) の使用
- SQL Server のソースとしての使用

データウェアハウスソース

- ソースとしての Amazon Redshift の使用
- ソースとしての Azure Synapse Analytics の使用
- ソースとして BigQuery を使用する
- ソースとしての Greenplum データベースの使用
- ソースとしての Netezza の使用
- ソースとしての Oracle データウェアハウスの使用
- ソースとしての Snowflake の使用
- SQL Server データウェアハウスをソースとしての使用
- ソースとしての Teradata の使用

- ソースとしての Vertica の使用

ビッグデータソース

- Apache Hadoop をソースとして使用
- Apache Oozie をソースとしての使用

AWS SCT で Amazon RDS と Amazon Aurora 接続を暗号化

アプリケーションから Amazon RDS または Amazon Aurora データベースへの暗号化された接続を開くには、AWS ルート証明書を何らかの形式のキーストレージにインポートする必要があります。ルート証明書は、Amazon RDS ユーザーガイド内の SSL/TLS を使用した DB インスタンスへの接続の暗号化にある AWS からダウンロードできます。

すべての AWS リージョンで使用できるルート証明書と、旧新両方のルート証明書が含まれる証明書バンドルの 2 つのオプションを使用できます。

使用する方法に応じて、次の 2 つの手順のうちのいずれかのステップに従います。

証明書を Windows システムストレージにインポートする

1. 次のいずれかのソースから証明書をダウンロードします。

   証明書のダウンロード方法の詳細については、『Amazon RDS ユーザーガイド』の「SSL/TLS を使用した DB インスタンス接続の暗号化」をご参照ください。
2. Windows の検索ウィンドウに、**`Manage computer certificates`** と入力します。コンピュータに変更を加えることをアプリケーションに許可するかどうかを尋ねるメッセージが表示されたら、はいを選択します。
3. 証明書のウィンドウが開いたら、必要に応じて [認定済み-ローカルコンピュータ] を展開し、証明書のリストが表示されるようにします。[Trusted Root Certification Authorities] (信頼されたルート証明機関) のコンテキスト (右クリック) メニューを開き、[All Tasks] (すべてのタスク)、[Import] (インポート) の順に選択します。
4. [Next] (次へ)、次に [Browse] (参照) を選択し、ステップ 1 でダウンロードした `*.pem` ファイルを見つけます。[Open] (開く) を選択して証明書ファイルを選択したら、[Next] (次へ) を選択し、[Finish] (終了) を選択します。

> **Note**
>
> ファイルを検索するには、参照ウィンドウで、ファイルタイプを [All files (*.*)] (すべてのファイル (*.*)) に変更する必要があります。これは、.pem は証明書の標準の拡張子ではないためです。

5. Microsoft マネジメントコンソールで、[Certificates] (証明書) を展開します。次に [Trusted Root Certification Authorities] (信頼されたルート証明機関) を展開して [Certificates] (証明書) を選択し、証明書を検索して存在することを確認します。証明書の名前は Amazon RDS で始まります。
6. コンピュータを再起動します。

証明書を Java キーストアにインポートする

1. 次のいずれかのソースから証明書をダウンロードします。

   証明書のダウンロード方法の詳細については、『Amazon RDS ユーザーガイド』の「SSL/TLS を使用した DB インスタンス接続の暗号化」をご参照ください。
2. 証明書バンドルをダウンロードした場合は、個々の証明書ファイルに分割します。分割するには、-----BEGIN CERTIFICATE----- で始まり -----END CERTIFICATE----- で終わる各証明書ブロックを個別の *.pem ファイルに配置します。各証明書に対して個別の *.pem ファイルを作成した後、証明書バンドルファイルは安全に削除できます。
3. 証明書をダウンロードしたディレクトリにあるコマンドウィンドウまたはターミナルセッションを開き、前のステップで作成した各 *.pem ファイルに対して次のコマンドを実行します。

   ```
   keytool -importcert -file <filename>.pem -alias <filename>.pem -keystore storename
   ```

   Example

   次の例では、eu-west-1-bundle.pem ファイルをダウンロード済みであることを前提としています。

   ```
   keytool -importcert -file eu-west-1-bundle.pem -alias eu-west-1-bundle.pem -
   keystore trust-2019.ks
   Picked up JAVA_TOOL_OPTIONS: -Dlog4j2.formatMsgNoLookups=true
   Enter keystore password:
   Re-enter new password:
   ```

```
Owner: CN=Amazon RDS Root 2019 CA, OU=Amazon RDS, O="Amazon Web Services, Inc.",
 ST=Washington, L=Seattle, C=US
Issuer: CN=Amazon RDS Root 2019 CA, OU=Amazon RDS, O="Amazon Web Services, Inc.",
 ST=Washington, L=Seattle, C=US
Serial number: c73467369250ae75
Valid from: Thu Aug 22 19:08:50 CEST 2019 until: Thu Aug 22 19:08:50 CEST 2024
Certificate fingerprints:
         SHA1: D4:0D:DB:29:E3:75:0D:FF:A6:71:C3:14:0B:BF:5F:47:8D:1C:80:96
         SHA256:
 F2:54:C7:D5:E9:23:B5:B7:51:0C:D7:9E:F7:77:7C:1C:A7:E6:4A:3C:97:22:E4:0D:64:54:78:FC:70:AA:
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 73 5F 60 D8 BC CB 03 98   F4 2B 17 34 2E 36 5A A6  s_`......+.4.6Z.
0010: 60 FF BC 1F                                        `...
]
]

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  Key_CertSign
  Crl_Sign
]

#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 73 5F 60 D8 BC CB 03 98   F4 2B 17 34 2E 36 5A A6  s_`......+.4.6Z.
0010: 60 FF BC 1F                                        `...
]
]
```

```
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

4. キーストアを AWS SCT の信頼ストアとして追加します。追加するには、メインメニューから [設定]、[全般設定]、[セキュリティ]、[信頼ストア] の順に選択し、[既存の信頼ストアを選択] を選択します。

   信頼ストアを追加した後は、データベースへの AWS SCT 接続を作成するときに SSL 対応接続を構成するために使用することができます。AWS SCT [Connect to database] (データベースに接続) ダイアログで、[Use SSL] (SSL の使用) をクリックし、以前に入力した信頼ストアを選択します。

Apache Cassandra を AWS SCT のソースとしての使用

AWS SCT を使用して、Apache Cassandra から Amazon DynamoDB にキースペースを変換することができます。

ソースとしての Apache Cassandra の接続

以下の手順を使用して、AWS Schema Conversion Tool で Apache Cassandra ソースデータベースに接続します。

Apache Cassandra ソースデータベースに接続する

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Cassandra]、次に [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。

- Apache Cassandra ソースデータベースの接続情報を手動で入力するには、次の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームサービス (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [信頼ストア]: 使用する信頼ストア。 • [キーストア]: 使用するキーストア。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Apache Hadoop を AWS SCT のソースとして使用

AWS SCT コマンドラインインターフェイス (CLI) を使用して Apache Hadoop から Amazon EMR に移行できます。AWS SCT は、Amazon S3 バケットを移行中のデータの一時ストレージとして使用します。

AWS SCT はバージョン 2.2.0 以降の Apache Hadoop のソースとしてサポートしています。また、AWS SCT はApache Hive バージョン 0.13.0 以降もサポートしています。

AWS SCT は Amazon EMR バージョン 6.3.0 以降をターゲットとしてサポートします。また、AWS SCT は Apache Hadoop バージョン 2.6.0 以降、および Apache Hive バージョン 0.13.0 以降をターゲットとしてサポートしています。

トピック

Apache Hadoop をソースとして使用する場合の前提条件

AWS SCT CLI を使用して Apache Hadoop に接続するには、次の前提条件が必要です。

- 移行中にデータを保存する Amazon S3 バケットを作成します。その後、データを Amazon EMR HDFS にコピーするか、Amazon S3 を Hadoop ワークロードのデータリポジトリとして使用できます。詳細については、Amazon S3 ユーザーガイドのバケットの作成を参照してください。
- `AmazonS3FullAccess` ポリシーを使用して AWS Identity and Access Management (IAM) ロールを作成します。AWS SCT はこの IAM ロールを使用して、Amazon S3 バケットにアクセスします。

- AWS シークレットキーと AWS シークレットアクセスキーをメモしておいてください。AWS アクセスキーの詳細については、『IAM ユーザーガイド』の「アクセスキーの管理」を参照してください。
- ターゲット Amazon EMR クラスターを作成して設定します。詳細については、『Amazon EMR 管理ガイド』の「Amazon EMR の使用開始」を参照してください。
- distcp ユーティリティをソース Apache Hadoop クラスターにインストールします。また、ターゲットの Amazon EMR クラスターに s3-dist-cp ユーティリティをインストールします。データベースユーザーがこれらのユーティリティを実行する権限を持っていることを確認してください。
- ソース Hadoop クラスター内の core-site.xml ファイルを s3a プロトコルを使用するように設定します。これを行うには、fs.s3a.aws.credentials.provider パラメータを次のいずれかの値に設定します。
  - org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider
  - org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider
  - org.apache.hadoop.fs.s3a.AnonymousAWSCredentialsProvider
  - org.apache.hadoop.fs.s3a.auth.AssumedRoleCredentialProvider

  次のコード例を core-site.xml ファイルに追加できます。

```
<property>
  <name>fs.s3a.aws.credentials.provider</name>
  <value>org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider</value>
</property>
```

  前の例は、前述のオプションリストにある 4 つのオプションのうち 1 つを示しています。core-site.xml ファイルに fs.s3a.aws.credentials.provider パラメータを設定しない場合、AWS SCT でプロバイダーが自動的に選択されます。

ソースとして Hive を使用する権限

Hive ソースユーザーに必要な権限は次のとおりです。

- ソースデータフォルダとソース Amazon S3 バケットへの READ アクセス
- 中間およびターゲットの Amazon S3 バケットへの READ+WRITE アクセス

移行速度を上げるために、ACID トランザクションソーステーブルのコンパクションを実行することをお勧めします。

Amazon EMR Hive のターゲットユーザーに必要な権限は次のとおりです。

- Amazon S3 バケットへの READ アクセス
- 中間 Amazon S3 バケットへの READ+WRITE アクセス
- ターゲット HDFS フォルダへの READ+WRITE アクセス

ソースとして HDFS を使用する権限

HDFS をソースとして使用するのに必要な権限は次のとおりです。

- NameNode 用の EXECUTE
- 移行プロジェクトに含める EXECUTE+READ のすべてのソースフォルダとファイル用
- Amazon S3 への移行前に Spark ジョブを実行し、ファイルを保存するための NameNode 内の tmp ディレクトリの READ+WRITE 用

HDFS では、すべての操作にトラバーサルアクセス権限が必要です。トラバーサルアクセスには、最後のパスコンポーネントを除いて、パスの既存のすべてのコンポーネントに対する EXECUTE アクセス権限が必要です。たとえば、/foo/bar/baz にアクセスする操作を行う場合、ユーザーには、EXECUTE、/、/foo、/foo/bar に対するアクセス権限が必要です。

次のコード例は、ソースフォルダとファイルへの EXECUTE+READ アクセス権限、および tmp ディレクトリへの READ+WRITE アクセス権限を付与する方法を示しています。

```
hadoop fs -chmod -R 744 /user/hdfs-data
hadoop fs -chmod -R 766 /tmp
```

HDFS をターゲットとして使用するためのアクセス権限

Amazon EMR Hive のターゲットユーザーに必要なアクセス権限は次のとおりです。

- ターゲット Amazon EMR クラスターの NameNode 用の EXECUTE
- 移行後にデータを保存するターゲット HDFS フォルダ用の READ+WRITE

ソースとしての Apache Hadoop への接続

AWS SCT バージョン 1.0.670 以降では、Apache Hadoop をソースとして使用できます。Hadoop クラスターは、AWS SCT コマンドラインインターフェイス (CLI) でのみ移行できます。開始する前に、AWS SCT のコマンドラインインターフェイスに慣れておきかしょう。詳細については、「AWS SCT CLI リファレンス」を参照してください。

AWS SCT CLI で Apache Hadoop に接続するには

1. 新しい AWS SCT CLI スクリプトを作成するか、既存のシナリオテンプレートを編集します。例えば、HadoopMigrationTemplate.scts テンプレートをダウンロードして編集できます。詳細については、「CLI シナリオの取得」を参照してください。
2. ドライバの場所やログフォルダなどの AWS SCT アプリケーション設定を行います。

   必要な JDBC ドライバーをダウンロードし、ファイルを保存する場所を指定します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。

   次のコード例では、Apache Hive ドライバーにパスを追加する方法を示します。このコード例を実行すると、AWS SCT はログファイルを c:\sct フォルダに保存します。

   ```
   SetGlobalSettings
       -save: 'true'
       -settings: '{
           "hive_driver_file": "c:\\sct\\HiveJDBC42.jar",
           "log_folder": "c:\\sct",
           "console_log_folder": "c:\\sct"
       }'
   /
   ```

   Windows では、この例と以下の例を使用できます。
3. 新しい AWS SCT プロジェクトを作成します。

   次のコード例では、c:\sct フォルダに hadoop_emr プロジェクトを作成します。

   ```
   CreateProject
       -name: 'hadoop_emr'
       -directory: 'c:\sct'
   /
   ```
4. ソース Hadoop クラスターをプロジェクトに追加します。

AddSourceCluster コマンドを使用して、ソース Hadoop クラスターに接続します。name、host、port、user の必須パラメータには必ず値を指定してください。その他のパラメータは省略可能です。

次のコード例では、ソース Hadoop クラスターを追加します。この例では、HADOOP_SOURCE をソースクラスターの名前として設定しています。このオブジェクト名を使用して Hive と HDFS サービスをプロジェクトに追加し、マッピングルールを作成します。

```
AddSourceCluster
    -name: 'HADOOP_SOURCE'
    -vendor: 'HADOOP'
    -host: 'hadoop_address'
    -port: '22'
    -user: 'hadoop_user'
    -password: 'hadoop_password'
    -useSSL: 'true'
    -privateKeyPath: 'c:\path\name.pem'
    -passPhrase: 'hadoop_passphrase'
/
```

前の例では、*hadoop_address* を Hadoop クラスターの IP アドレスに置き換えてください。必要に応じて port オプションの値を設定します。次に、*hadoop_user* と *hadoop_password* を Hadoop ユーザーの名前とこのユーザーのパスワードに置き換えます。*path\name* には、ソース Hadoop クラスターの PEM ファイルの名前とパスを入力します。

5. CLI スクリプトを保存します。次に、Hive と HDFS サービスの接続情報を追加します。

Hive と HDFS のソースサービスへの接続

AWS SCT CLI を使用して、ソース Hive および HDFS サービスに接続できます。Apache Hive に接続するには、Hive JDBC ドライバーバージョン 2.3.4 以降を使用してください。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。

AWS SCT は hadoop クラスターユーザーを使用して Apache Hive に接続します。これを行うには、AddSourceClusterHive および AddSourceClusterHDFS コマンドを使用します。これを行うには、次のいずれかのアプローチを使用できます。

- 新しい SSH トンネルを作成します。

createTunnel に「**true**」と入力します。host には、ソース Hive または HDFS サービスの内部 IP アドレスを入力します。port には、Hive または HDFS サービスのサービスポートを入力します。

次に、user および password の Hive または HDFS の認証情報を入力します。SSH トンネルの詳細については、『Amazon EMR 管理ガイド』の「ローカルポート転送を使用したプライマリノードへの SSH トンネルのセットアップ」を参照してください。

- 既存の SSH トンネルを使用する。

  host に「**localhost**」と入力します。port には、SSH トンネルパラメータからローカルポートを入力します。

- Hive および HDFS サービスに直接接続します。

  host には、ソース Hive または HDFS サービスの IP アドレスまたはホスト名を入力します。port には、Hive または HDFS サービスのサービスポートを入力します。次に、user および password の Hive または HDFS の認証情報を入力します。

AWS SCT CLI で Hive と HDFS に接続するには

1. ソース Hadoop クラスターの接続情報を含む CLI スクリプトを開きます。前のステップで定義した Hadoop クラスターの名前を使用してください。
2. ソース Hive サービスをプロジェクトに追加します。

   AddSourceClusterHive コマンドを使用してソース Hive サービスを接続します。user、password、cluster、name、port の必須パラメータには必ず値を指定してください。その他のパラメータは省略可能です。

   次のコード例では、Hive サービスと連携するための AWS SCT のトンネルを作成します。このソース Hive サービスは AWS SCT と同じ PC 上で動作します。この例では、前の例の HADOOP_SOURCE ソースクラスターを使用しています。

```
AddSourceClusterHive
    -cluster: 'HADOOP_SOURCE'
    -name: 'HIVE_SOURCE'
    -host: 'localhost'
    -port: '10005'
    -user: 'hive_user'
    -password: 'hive_password'
```

```
    -createTunnel: 'true'
    -localPort: '10005'
    -remoteHost: 'hive_remote_address'
    -remotePort: 'hive_port'
/
```

次のコード例は、トンネルなしで Hive サービスに接続します。

```
AddSourceClusterHive
    -cluster: 'HADOOP_SOURCE'
    -name: 'HIVE_SOURCE'
    -host: 'hive_address'
    -port: 'hive_port'
    -user: 'hive_user'
    -password: 'hive_password'
/
```

前述の例では、*hive_user* と *hive_password* を Hive ユーザーの名前とこのユーザーのパスワードに置き換えます。

次に、*hive_address* と *hive_port* をソース Hadoop クラスターNameNode IP アドレスとポートに置き換えます。

hive_remote_address には、ソースの Hive サービスのデフォルト値 `127.0.0.1` または NameNode IP アドレスを使用できます。

3. ソース HDFS サービスをプロジェクトに追加します。

   `AddSourceClusterHDFS` コマンドを使用してソース HDFS サービスを接続します。`user`、`password`、`cluster`、`name`、`port` の必須パラメータには必ず値を指定してください。その他のパラメータは省略可能です。

   ソース HDFS サービスからデータを移行するために必要なアクセス権限がユーザーにあることを確認します。詳細については、「ソースとして Hive を使用する権限」を参照してください。

   次のコード例では、Apache HDFS サービスと連携するための AWS SCT のトンネルを作成します。この例では、前に作成した `HADOOP_SOURCE` ソースクラスターを使用しています。

```
AddSourceClusterHDFS
    -cluster: 'HADOOP_SOURCE'
    -name: 'HDFS_SOURCE'
```

```
    -host: 'localhost'
    -port: '9005'
    -user: 'hdfs_user'
    -password: 'hdfs_password'
    -createTunnel: 'true'
    -localPort: '9005'
    -remoteHost: 'hdfs_remote_address'
    -remotePort: 'hdfs_port'
/
```

次のコードはトンネルなしで Apache HDFS サービスに接続します。

```
AddSourceClusterHDFS
    -cluster: 'HADOOP_SOURCE'
    -name: 'HDFS_SOURCE'
    -host: 'hdfs_address'
    -port: 'hdfs_port'
    -user: 'hdfs_user'
    -password: 'hdfs_password'
/
```

前述の例では、*hdfs_user* と *hdfs_password* を HDFS ユーザーの名前とこのユーザーのパスワードに置き換えます。

次に、*hdfs_address* と *hdfs_port* をソース Hadoop クラスターNameNode IP アドレスとポートに置き換えます。

hdfs_remote_address には、ソースの Hive サービスのデフォルト値 `127.0.0.1` または NameNode IP アドレスを使用できます。

4. CLI スクリプトを保存します。次に、ターゲット Amazon EMR クラスターの接続情報と移行コマンドを追加します。

Amazon EMR にターゲットとして接続する

AWS SCT CLI を使用してターゲット Amazon EMR クラスターに接続できます。そのためには、インバウンドトラフィックを承認し、SSH を使用します。この場合、AWS SCT には Amazon EMR クラスターを操作するために必要なすべてのアクセス権限があります。詳細については、『Amazon EMR 管理ガイド』の「接続する前に」と「SSH を使用してプライマリノードに接続する」を参照してください。

AWS SCT は hadoop クラスターユーザーを使用して Amazon EMR ハイブに接続します。Amazon EMR Hive に接続するには、Hive JDBC ドライバーバージョン 2.6.2.1002 以降を使用してください。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。

AWS SCT CLI で Amazon EMR に接続するには

1. ソース Hadoop クラスターの接続情報を含む CLI スクリプトを開きます。ターゲットの Amazon EMR 認証情報をこのファイルに追加します。
2. ターゲットの Amazon EMR クラスターをプロジェクトに追加します。

   次のコード例では、ターゲットの Amazon EMR クラスターを追加します。この例では、HADOOP_TARGET をターゲットクラスターの名前として設定します。このオブジェクト名を使用して、Hive と HDFS サービス、および Amazon S3、バケットフォルダをプロジェクトに追加し、マッピングルールを作成します。

   ```
   AddTargetCluster
    -name: 'HADOOP_TARGET'
    -vendor: 'AMAZON_EMR'
    -host: 'ec2-44-44-55-66.eu-west-1.EXAMPLE.amazonaws.com'
    -port: '22'
    -user: 'emr_user'
    -password: 'emr_password'
    -useSSL: 'true'
    -privateKeyPath: 'c:\path\name.pem'
    -passPhrase: '1234567890abcdef0!'
    -s3Name: 'S3_TARGET'
    -accessKey: 'AKIAIOSFODNN7EXAMPLE'
    -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
    -region: 'eu-west-1'
    -s3Path: 'doc-example-bucket/example-folder'
   /
   ```

   前の例では、AWS リソース名と Amazon EMR 接続情報を入力します。これには、Amazon EMR クラスターの IP アドレス、AWS アクセスキー、AWS シークレットアクセスキー、Amazon S3 バケットが含まれます。必要に応じて、ポート変数の値を設定します。次に、*emr_user* と *emr_password* を Amazon EMR ユーザーの名前とこのユーザーのパスワードに置き換えます。*path\ n name* には、ターゲット Amazon EMR クラスターの PEM ファイルの名前とパスを入力します。詳細については、「EMR クラスターアクセス用の PEM ファイルのダウンロード」を参照してください。
3. ターゲット Amazon S3 バケットをプロジェクトに追加します。

次のコード例では、ターゲットの Amazon S3 バケットを追加します。この例では、前に作成した HADOOP_TARGET クラスターを使用しています。

```
AddTargetClusterS3
 -cluster: 'HADOOP_TARGET'
 -Name: 'S3_TARGET'
 -accessKey: 'AKIAIOSFODNN7EXAMPLE'
 -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
 -region: 'eu-west-1'
 -s3Path: 'doc-example-bucket/example-folder'
/
```

前述の例では、AWS アクセスキー、AWS シークレットアクセスキー、および Amazon S3 バケットを入力します。

4. ターゲットの Hive サービスをプロジェクトに追加します。

次のコード例では、ターゲットの Hive サービスと連携するために AWS SCT のトンネルを作成します。この例では、前に作成した HADOOP_TARGET ターゲットクラスターを使用しています。

```
AddTargetClusterHive
    -cluster: 'HADOOP_TARGET'
    -name: 'HIVE_TARGET'
    -host: 'localhost'
    -port: '10006'
    -user: 'hive_user'
    -password: 'hive_password'
    -createTunnel: 'true'
    -localPort: '10006'
    -remoteHost: 'hive_address'
    -remotePort: 'hive_port'
/
```

前述の例では、*hive_user* と *hive_password* を Hive ユーザーの名前とこのユーザーのパスワードに置き換えます。

次に、*hive_address* をデフォルト値 127.0.0.1 またはターゲットの Hive サービスの NameNode IP アドレスに置き換えます。次に、*hive_port* を対象の Hive サービスのポートに置き換えます。

5. ターゲット HDFS サービスをプロジェクトに追加します。

   次のコード例では、Apache HDFS サービスと連携するための AWS SCT のトンネルを作成します。この例では、前に作成した HADOOP_TARGET ターゲットクラスターを使用しています。

```
AddTargetClusterHDFS
    -cluster: 'HADOOP_TARGET'
    -name: 'HDFS_TARGET'
    -host: 'localhost'
    -port: '8025'
    -user: 'hdfs_user'
    -password: 'hdfs_password'
    -createTunnel: 'true'
    -localPort: '8025'
    -remoteHost: 'hdfs_address'
    -remotePort: 'hdfs_port'
/
```

   前述の例では、*hdfs_user* と *hdfs_password* を HDFS ユーザーの名前とこのユーザーのパスワードに置き換えます。

   次に、*hdfs_address* と *hdfs_port* を、ターゲット HDFS サービスのプライベート IP アドレスとNameNode ポートに置き換えます。

6. CLI スクリプトを保存します。次に、マッピングルールと移行コマンドを追加します。詳細については、「Apache Hadoop から Amazon EMR への移行」を参照してください。

Apache Oozie を AWS SCT のソースとしての使用

AWS SCT コマンドラインインターフェイス (CLI) を使用して、Apache Oozie ワークフローを AWS Step Functions に変換できます。Apache Hadoop ワークロードを Amazon EMR に移行したら、AWS クラウド のネイティブサービスを使用してジョブをオーケストレーションできます。詳細については、「Apache Hadoop をソースとして使用」を参照してください。

AWS SCT は、Oozie ワークフローを AWS Step Functions に変換し、AWS Lambda を使用して AWS Step Functions がサポートしていない機能をエミュレートします。また、Oozie AWS SCT ジョブのプロパティを AWS Systems Manager に変換します。

Apache Oozie ワークフローを変換するには、必ず AWS SCT バージョン 1.0.671 以降を使用してください。また、AWS SCT のコマンドラインインターフェイスにも慣れておいてください。詳細については、「AWS SCT CLI リファレンス」を参照してください。

Apache Oozie をソースとして使用する場合の前提条件

AWS SCT CLI を使用して Apache Oozie に接続するには、以下の前提条件を満たす必要があります。

- ステートマシンの定義を保存する Amazon S3 バケットを作成します。これらの定義を使用してステートマシンを設定できます。詳細については、Amazon S3 ユーザーガイドのバケットの作成を参照してください。
- `AmazonS3FullAccess` ポリシーを使用して AWS Identity and Access Management (IAM) ロールを作成します。AWS SCT はこの IAM ロールを使用して、Amazon S3 バケットにアクセスします。
- AWS シークレットキーと AWS シークレットアクセスキーをメモしておいてください。AWS アクセスキーの詳細については、『IAM ユーザーガイド』の「アクセスキーの管理」を参照してください。
- AWS 認証情報と Amazon S3 バケットに関する情報を、グローバルアプリケーション設定の AWS サービスプロファイルに保存します。次に、AWS SCT は この AWS サービスプロファイルを使用して AWS リソースを操作します。詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。

AWS SCT でソース Apache Oozie ワークフローを使用するには、ソースファイルの特定の構造が必要です。各アプリケーションフォルダには `job.properties` ファイルが含まれている必要があります。このファイルには、ジョブプロパティのキーと値のペアが含まれています。また、各アプリケーションフォルダにこの `workflow.xml` ファイルが含まれている必要があります。このファイルには、ワークフローのアクションノードと制御フローノードが記述されています。

ソースとしての Apache Cassandra への接続

以下の手順を使用して、Apache Oozie ソースファイルに接続します。

AWS SCT CLI で Apache Oozie に接続するには

1. 新しい AWS SCT CLI スクリプトを作成するか、既存のシナリオテンプレートを編集します。例えば、`OozieConversionTemplate.scts` テンプレートをダウンロードして編集できます。詳細については、「CLI シナリオの取得」を参照してください。

2. AWS SCT アプリケーション設定を構成します。

   次のコード例では、アプリケーション設定を保存し、プロジェクトにパスワードを保存できます。保存した設定は他のプロジェクトでも使用できます。

```
SetGlobalSettings
    -save: 'true'
    -settings: '{
        "store_password": "true"
    }'
/
```

3. 新しい AWS SCT プロジェクトを作成します。

   次のコード例では、c:\sct フォルダに oozie プロジェクトを作成します。

```
CreateProject
    -name: 'oozie'
    -directory: 'c:\sct'
/
```

4. AddSource コマンドを使用して、ソース Apache Oozie ファイルを含むフォルダをプロジェクトに追加します。vendor パラメータの APACHE_OOZIE 値は必ず使用してください。以下の必須のパラメータの値 name および mappingsFolder を指定します。

   次のコード例では、Apache Oozie をソースとして AWS SCT プロジェクトに追加しています。この例では、OOZIE という名前のソースオブジェクトを作成します。このオブジェクト名を使用してマッピングルールを追加します。このコード例を実行すると、AWS SCT は c:\oozie フォルダを使用してプロジェクトにソースファイルを読み込みます。

```
AddSource
    -name: 'OOZIE'
    -vendor: 'APACHE_OOZIE'
    -mappingsFolder: 'c:\oozie'
/
```

   Windows では、この例と以下の例を使用できます。

5. ConnectSource コマンドを使用してソース Apache Oozie ファイルに接続します。前のステップで定義されたソースオブジェクトの名前を使用します。

```
ConnectSource
    -name: 'OOZIE'
    -mappingsFolder: 'c:\oozie'
/
```

6. CLI スクリプトを保存します。次に、AWS Step Functions サービスの接続情報を追加します。

拡張パック内の AWS Lambda 関数を使用するための権限

AWS Step Functions がサポートしていないソース関数については、AWS SCT が拡張パックを作成します。この拡張パックには、ソース関数をエミュレートする AWS Lambda 関数が含まれています。

この拡張パックを使用するには、以下の権限を持つ AWS Identity and Access Management (IAM) ロールを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "lambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:498160209112:function:LoadParameterInitialState:*",
                "arn:aws:lambda:*:498160209112:function:EvaluateJSPELExpressions:*"
            ]
        },
        {
            "Sid": "emr",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:AddJobFlowSteps"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:*:498160209112:cluster/*"
            ]
        },
```

```
        {
            "Sid": "s3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::*/*"
            ]
        }
    ]
}
```

エクステンションパックを適用するには、AWS SCT に次のアクセス権限を持つ IAM ロールが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListRolePolicies",
                "iam:CreateRole",
                "iam:TagRole",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:DeleteRole",
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::ACCOUNT_NUMBER:role/sct/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListRolePolicies"
            ],
            "Resource": [
```

```
                "arn:aws:iam::ACCOUNT_NUMBER:role/
lambda_LoadParameterInitialStateRole",
                "arn:aws:iam::ACCOUNT_NUMBER:role/lambda_EvaluateJSPELExpressionsRole",
                "arn:aws:iam::ACCOUNT_NUMBER:role/
stepFunctions_MigratedOozieWorkflowRole"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction",
                "lambda:CreateFunction",
                "lambda:UpdateFunctionCode",
                "lambda:DeleteFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:ACCOUNT_NUMBER:function:LoadParameterInitialState",
                "arn:aws:lambda:*:ACCOUNT_NUMBER:function:EvaluateJSPELExpressions"
            ]
        }
    ]
}
```

AWS Step Functions にターゲットとしてに接続する

以下の手順を使用して、ターゲットとして AWS Step Functions に接続します。

AWS SCT CLI で AWS Step Functions に接続するには

1. Apache Oozie ソースファイルの接続情報を含む CLI スクリプトを開きます。
2. AddTarget コマンドを使用して、移行ターゲットに関する情報を AWS SCT プロジェクトに追加します。vendor パラメータの STEP_FUNCTIONS 値は必ず使用してください。以下の必須のパラメータの値 name および profile を指定します。

   次のコード例は、AWS Step Functions を AWS SCT プロジェクトにソースとして追加します。この例では、AWS_STEP_FUNCTIONS という名前のターゲットオブジェクトを作成します。マッピングルールを作成するときには、このオブジェクト名を使用してください。また、この例では、前提条件のステップで作成した AWS SCT サービスプロファイルを使用しています。*profile_name* は必ず自分のプロファイルの名前に置き換えてください。

   ```
   AddTarget
   ```

```
    -name: 'AWS_STEP_FUNCTIONS'
    -vendor: 'STEP_FUNCTIONS'
    -profile: 'profile_name'
/
```

AWS サービスプロファイルを使用しない場合は、accessKey、secretKey、awsRegion、s3Path の必須パラメータの値を指定してください。これらのパラメータを使用して、AWS シークレットアクセスキー、AWS シークレットキー、AWS リージョン、Amazon S3 バケットへのパスを指定します。

3. ConnectTarget コマンドを使用して AWS Step Functions に接続します。前のステップで定義されたターゲットオブジェクトの名前を使用します。

   次のコード例では、AWS サービスプロファイルを使用して AWS_STEP_FUNCTIONS ターゲットオブジェクトに接続します。*profile_name* は必ず自分のプロファイルの名前に置き換えてください。

   ```
   ConnectTarget
       -name: 'AWS_STEP_FUNCTIONS'
       -profile: 'profile_name'
   /
   ```

4. CLI スクリプトを保存します。次に、マッピングルールと移行コマンドを追加します。詳細については、「Apache Oozie を AWS Step Functions に変換 する」を参照してください。

Azure SQL データベースの AWS SCT のソースとしての使用

AWS SCT を使用して、Azure SQL からのスキーマ、コードオブジェクトおよびアプリケーションコードを次のターゲットに変換できます。

- Amazon RDS for MySQL
- Amazon Aurora MySQL 互換エディション
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション

トピック

- ソースとしての Azure SQL データベースの権限
- ソースとしての Azure SQL データベースへの接続

ソースとしての Azure SQL データベースの権限

ソースとしての Azure SQL データベースに必要な権限を以下に示します。

- VIEW DEFINITION
- VIEW DATABASE STATE

スキーマを変換する、各データベースの付与を繰り返します。

ターゲットの MySQL および PostgreSQL データベースに必要な権限については、以下のセクションで説明します。

- MySQL をターゲットデータベースとする場合の権限
- ターゲットデータベースとしての PostgreSQL の権限

ソースとしての Azure SQL データベースへの接続

以下の手順を使用して、AWS Schema Conversion Tool で Azure SQL データベースのソースデータベースに接続します。

Azure SQL データベースのソースデータベースへの接続

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Azure SQL Database] (Azure SQL データベース) を選択し、次に [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。

- Azure SQL Database のソースデータベース接続情報を、手動で入力するには、次の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームサービス (DNS) 名または IP アドレスを入力します。
データベース	接続するデータベースのデータベース名を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

IBM Db2 for z/OS を AWS SCT のソースとして使用する

AWS SCT を使用して、IBM Db2 for z/OS からのスキーマ、コードオブジェクトおよびアプリケーションコードを次のターゲットに変換できます。

- Amazon RDS for MySQL

- Amazon Aurora MySQL 互換エディション
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション

Db2 for z/OS をソースデータベースとして使用するための前提条件

IBM Db2 for z/OS バージョン 12 ファンクションレベル 100 のデータベースバージョンは、IBM Db2 for z/OS バージョン 12 の新機能のほとんどをサポートしていません。このデータベースバージョンは、Db2 バージョン 11 へのフォールバックと Db2 バージョン 11 とのデータ共有をサポートしています。Db2 バージョン 11 のサポートされていない機能の変換を避けるため、IBM Db2 for z/OS データベース機能レベル 500 以上を AWS SCT のソースとして使用することをお勧めします。

次のコード例を使用して、ソース IBM Db2 for z/OS データベースのバージョンを確認できます。

```
SELECT GETVARIABLE('SYSIBM.VERSION') as version FROM SYSIBM.SYSDUMMY1;
```

このコードがバージョン DSN12015 以上を返すことを確認してください。

次のコード例を使用して、ソース IBM Db2 for z/OS データベース内の APPLICATION COMPATIBILITY 特殊レジスタの値を確認できます。

```
SELECT CURRENT APPLICATION COMPATIBILITY as version FROM SYSIBM.SYSDUMMY1;
```

このコードがバージョン V12R1M500 以上を返すことを確認してください。

ソースデータベースとしての Db2 for z/OS の権限

Db2 for z/OS データベースに接続し、システムカタログとテーブルを読み取るのに必要な権限は次のとおりです。

- SELECT ON SYSIBM.LOCATIONS
- SELECT ON SYSIBM.SYSCHECKS
- SELECT ON SYSIBM.SYSCOLUMNS
- SELECT ON SYSIBM.SYSDATABASE
- SELECT ON SYSIBM.SYSDATATYPES

- SELECT ON SYSIBM.SYSDUMMY1
- SELECT ON SYSIBM.SYSFOREIGNKEYS
- SELECT ON SYSIBM.SYSINDEXES
- SELECT ON SYSIBM.SYSKEYCOLUSE
- SELECT ON SYSIBM.SYSKEYS
- SELECT ON SYSIBM.SYSKEYTARGETS
- SELECT ON SYSIBM.SYSJAROBJECTS
- SELECT ON SYSIBM.SYSPACKAGE
- SELECT ON SYSIBM.SYSPARMS
- SELECT ON SYSIBM.SYSRELS
- SELECT ON SYSIBM.SYSROUTINES
- SELECT ON SYSIBM.SYSSEQUENCES
- SELECT ON SYSIBM.SYSSEQUENCESDEP
- SELECT ON SYSIBM.SYSSYNONYMS
- SELECT ON SYSIBM.SYSTABCONST
- SELECT ON SYSIBM.SYSTABLES
- SELECT ON SYSIBM.SYSTABLESPACE
- SELECT ON SYSIBM.SYSTRIGGERS
- SELECT ON SYSIBM.SYSVARIABLES
- SELECT ON SYSIBM.SYSVIEWS

Db2 for z/OS テーブルを PostgreSQL パーティションテーブルに変換するには、次に示す RUNSTATS ユーティリティを使用してデータベース内のテーブルスペースとテーブルに関する統計を収集します。

```
LISTDEF YOURLIST INCLUDE TABLESPACES DATABASE YOURDB
RUNSTATS TABLESPACE
LIST YOURLIST
TABLE (ALL) INDEX (ALL KEYCARD)
UPDATE ALL
REPORT YES
SHRLEVEL REFERENCE
```

前述の例では、*YOURDB* プレースホルダーをソースデータベースの名前に置き換えます。

ソースとしての Db2 for z/OS への接続

以下の手順を使用して、Db2 for z/OS ソースデータベースを AWS SCT に接続します。

IBM Db2 for z/OS ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Db2 for z/OS]、[次へ] の順に選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - IBM Db2 for z/OS ソースデータベース接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
場所	アクセスする Db2 ロケーションの一意名を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。

パラメータ	アクション
	AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。この場所を [グローバル設定] に追加すると、ここに表示されます。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[Db2 for z/OS ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

MySQL をターゲットデータベースとする場合の権限

ターゲットとして MySQL に必要な権限を以下に示します。

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- SELECT ON mysql.proc
- INSERT, UPDATE ON AWS_DB2ZOS_EXT.*
- INSERT, UPDATE, DELETE ON AWS_DB2ZOS_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_DB2ZOS_EXT_DATA.*

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
```

```
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT SELECT ON mysql.proc TO 'user_name';
GRANT INSERT, UPDATE ON AWS_DB2ZOS_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_DB2ZOS_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_DB2ZOS_EXT_DATA.* TO 'user_name';
```

上記の例の *user_name* は使用するユーザー名に置き換えます。次に、*your_password* を安全なパスワードに置き換えます。

Amazon RDS for MySQL をターゲットとして使用するには、log_bin_trust_function_creators パラメータを true に設定し、character_set_server を latin1 に設定します。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

Aurora MySQL をターゲットとして使用するには、log_bin_trust_function_creators パラメータを true に設定し、character_set_server を latin1 に設定します。lower_case_table_names パラメータを true に設定します。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

ターゲットデータベースとしての PostgreSQL の権限

PostgreSQL をターゲットとして使用するには、AWS SCT に CREATE ON DATABASE 権限が必要です。ターゲット PostgreSQL データベースごとにこの権限を必ず付与してください。

Amazon RDS for PostgreSQL AWS SCT をターゲットとして使用するには、rds_superuser 権限が必要です。

変換されたパブリックシノニムを使用するには、データベースのデフォルト検索パスを "$user", public_synonyms, public に変更します。

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
GRANT rds_superuser TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

上記の例の *user_name* は使用するユーザー名に置き換えます。[*db_name*] をターゲットデータベースの名前に置き換えます。最後に、[*your_password*] を安全なパスワードに置き換えます。

PostgreSQL では、スキーマを削除できるのはスキーマ所有者または superuser のみです。スキーマ所有者が一部のオブジェクトを所有していない場合でも、スキーマとスキーマに含まれるすべてのオブジェクトを削除できます。

異なるユーザーを使用して異なるスキーマを変換してターゲットデータベースに適用すると、AWS SCT で、スキーマを削除できないときにエラーメッセージが表示されることがあります。このエラーメッセージを回避するには、superuser ロールを使用してください。

Db2 for z/OS から PostgreSQL への変換設定

Db2 for z/OS to PostgreSQL の変換設定を編集するには、[設定] を選択し、次に [変換設定] を選択します。上のリストから [Db2 for z/OS] を選択し、次に [Db2 for z/OS – PostgreSQL] または [Db2 for z/OS – Amazon Aurora (PostgreSQL 互換)] を選択します。AWS SCT で、IBM Db2 for z/OS から PostgreSQL への変換に使用できるすべての設定が表示されます。

AWS SCT の Db2 for z/OS から PostgreSQL への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- ターゲットデータベース内の制約に固有の名前を生成するには。

  PostgreSQL では、使用するすべての制約名は一意でなければなりません。AWS SCT では、制約の名前にテーブル名のプレフィックスを追加することで、変換されたコード内の制約に固有の名前を生成できます。AWS SCT ですべての制約に固有の名前が生成されるようにするには、[制約の固有の名前を生成] を選択します。
- 変換されたコード内の DML ステートメントの列名、式、および句の書式設定を保持するには。

  AWS SCT は、DML ステートメント内の列名、式、句のレイアウトを、ソース コードと同様の位置と順序で保持できます。そのためには、[DML ステートメントの列名、式、句のフォーマットを保持] するで [はい] を選択します。
- テーブルパーティションを変換スコープから除外するには。

AWS SCT は、変換中にソーステーブルのすべてのパーティションをスキップできます。そのためには、[変換スコープからテーブルパーティションを除外する] を選択します。

- 増加に応じてパーティション化されたテーブルに自動パーティション化を使用するには。

  データ移行では、AWS SCT は、指定したサイズより大きいすべてのテーブルを自動的にパーティション分割できます。このオプションを使用するには、[より大きいテーブルのパーティションを強制する] を選択し、テーブルのサイズをギガバイト単位で入力します。次に、パーティション数を入力します。このオプションをオンにすると、AWS SCT でソースデータベースのダイレクトアクセスストレージデバイス (DASD) のサイズが考慮されます。

  AWS SCT はパーティション数を自動的に決定できます。そのためには、[パーティション数を比例的に増やす] を選択し、最大パーティション数を入力します。

- 動的結果セットを refcursor データ型の値の配列として返すには。

  AWS SCT は、動的結果セットを返すソースプロシージャを、追加の出力パラメータとしてオープンリフカーソル配列を持つプロシージャに変換できます。そのためには、[参照カーソルの配列を使ってすべての動的結果セットを返す] を選択します。

- 日付と時刻の値を文字列表現に変換する際に使用する標準を指定するには。

  AWS SCT は、サポートされている業界フォーマットのいずれかを使用して、日付と時刻の値を文字列表現に変換できます。そのためには、[日付値の文字列表現を使用する] または [時間値の文字列表現を使用する] を選択します。次に、以下のいずれかの標準を選択します。

  - 国際標準化機構 (ISO)
  - IBM 欧州規格 (EUR)
  - IBM 米国規格 (米国)
  - 日本工業規格西暦 (JIS)

IBM Db2 LUW を AWS SCT のソースとして使用する

AWS SCT を使用して、スキーマ、SQL 言語のコード オブジェクト、およびアプリケーション コードを IBM Db2 for Linux, Unix, and Windows (Db2 LUW) から次のターゲットに変換できます。

- Amazon RDS for MySQL
- Amazon Aurora MySQL 互換エディション
- Amazon RDS for PostgreSQL

- Amazon Aurora PostgreSQL 互換エディション
- Amazon RDS for MariaDB

AWS SCT では、ソースとして Db2 LUW バージョン 9.1、9.5、9.7、10.1、10.5、11.1、および 11.5 がサポートされています。

ソースとしての Db2 LUW の権限

Db2 LUW データベースに接続して利用可能な特権を確認し、ソースのスキーマメタデータを読み取るために必要な特権を以下に示します。

- 接続を確立するために必要な特権:
  - CONNECT ON DATABASE
- SQL ステートメントを実行するために必要な特権:
  - EXECUTE ON PACKAGE NULLID.SYSSH200
- インスタンスレベルの情報を取得するために必要な特権:
  - EXECUTE ON FUNCTION SYSPROC.ENV_GET_INST_INFO
  - SELECT ON SYSIBMADM.ENV_INST_INFO
  - SELECT ON SYSIBMADM.ENV_SYS_INFO
- ロール、グループ、および機関を通じて付与された特権を確認するために必要な特権:
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID
  - SELECT ON SYSIBMADM.PRIVILEGES
- システムカタログとテーブルで必要な特権:
  - SELECT ON SYSCAT.ATTRIBUTES
  - SELECT ON SYSCAT.CHECKS
  - SELECT ON SYSCAT.COLIDENTATTRIBUTES
  - SELECT ON SYSCAT.COLUMNS
  - SELECT ON SYSCAT.DATAPARTITIONEXPRESSION
  - SELECT ON SYSCAT.DATAPARTITIONS
  - SELECT ON SYSCAT.DATATYPEDEP
  - SELECT ON SYSCAT.DATATYPES

- SELECT ON SYSCAT.HIERARCHIES
- SELECT ON SYSCAT.INDEXCOLUSE
- SELECT ON SYSCAT.INDEXES
- SELECT ON SYSCAT.INDEXPARTITIONS
- SELECT ON SYSCAT.KEYCOLUSE
- SELECT ON SYSCAT.MODULEOBJECTS
- SELECT ON SYSCAT.MODULES
- SELECT ON SYSCAT.NICKNAMES
- SELECT ON SYSCAT.PERIODS
- SELECT ON SYSCAT.REFERENCES
- SELECT ON SYSCAT.ROUTINEPARMS
- SELECT ON SYSCAT.ROUTINES
- SELECT ON SYSCAT.ROWFIELDS
- SELECT ON SYSCAT.SCHEMATA
- SELECT ON SYSCAT.SEQUENCES
- SELECT ON SYSCAT.TABCONST
- SELECT ON SYSCAT.TABLES
- SELECT ON SYSCAT.TRIGGERS
- SELECT ON SYSCAT.VARIABLEDEP
- SELECT ON SYSCAT.VARIABLES
- SELECT ON SYSCAT.VIEWS
- SELECT ON SYSIBM.SYSDUMMY1

- SQL ステートメントを実行するには、データベースで有効になっているワークロードの少なくとも 1 つを使用する特権がユーザーアカウントに必要です。いずれのワークロードもユーザーに割り当てられていない場合は、デフォルトのユーザーワークロードがユーザーに対してアクセス可能であることを確認します。
  - ワークロードでの使用 (システム/デフォルト/ユーザーワークロード)

クエリを実行するには、ページサイズが 8K、16K、および 32K であるシステム一時テーブルスペースを作成する必要があります (存在しない場合)。一時テーブルスペースを作成するには、以下のスクリプトを実行します。

```
CREATE BUFFERPOOL BP8K
  IMMEDIATE
  ALL DBPARTITIONNUMS
  SIZE AUTOMATIC
  NUMBLOCKPAGES 0
  PAGESIZE 8K;

CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_8K
  PAGESIZE 8192
  BUFFERPOOL BP8K;

CREATE BUFFERPOOL BP16K
  IMMEDIATE
  ALL DBPARTITIONNUMS
  SIZE AUTOMATIC
  NUMBLOCKPAGES 0
  PAGESIZE 16K;

CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_BP16K
  PAGESIZE 16384
  BUFFERPOOL BP16K;

CREATE BUFFERPOOL BP32K
  IMMEDIATE
  ALL DBPARTITIONNUMS
  SIZE AUTOMATIC
  NUMBLOCKPAGES 0
  PAGESIZE 32K;

CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_BP32K
  PAGESIZE 32768
  BUFFERPOOL BP32K;
```

ソースとしての Db2 LUW への接続

以下の手順を使用して、Db2 LUW ソースデータベースを AWS Schema Conversion Tool に接続します。

Db2 LUW ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Db2 LUW]、[次へ] の順に選択します。

[Add source] (ソースの追加) ダイアログボックスが表示されます。

3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。

     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。

   - IBM Db2 LUW ソースデータベース接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
データベース	Db2 LUW データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。

パラメータ	アクション
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。この場所を [グローバル設定] に追加すると、ここに表示されます。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[DB2 LUW のドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Db2 LUW から Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL 互換エディションへの変換

IBM Db2 LUW を PostgreSQL に移行すると、AWS SCT は Db2 LUW で使用されるさまざまなトリガー文を変換できます。例えば、次のようなトリガーステートメントです。

- トリガーイベント - INSERT、DELETE、および UPDATE トリガーイベントは、イベントがサブジェクトテーブルまたはサブジェクトビューに適用されるたびに、トリガーされたアクションの実行を指定します。INSERT、DELETE、および UPDATE イベントの任意の組み合わせを指定で

きますが、各イベントを指定できるのは 1 回だけです。AWS SCT は、単一および複数のトリガーイベントをサポートします。イベントについては、PostgreSQL が実質的に同じ機能を備えています。

- イベント OF COLUMN - ベーステーブルからの列名を指定できます。このトリガーは、列名リストで列の更新が識別された場合にのみアクティブ化されます。PostgreSQL にも同じ機能があります。
- ステートメントトリガー – トリガーされるアクションがステートメント全体で 1 回のみ適用されるように指定します。このタイプのトリガー単位を、BEFORE トリガーまたは INSTEAD OF トリガーに指定することはできません。指定した場合、対象の行がなくても、UPDATE または DELETE トリガーがアクティブ化されます。PostgreSQL にもこの機能があり、ステートメントトリガーのトリガー宣言は PostgreSQL と Db2 LUW で同じです。
- Referencing 句 - 移行テーブルの移行変数とテーブル名の相関を指定します。相関名は、SQL オペレーションがトリガーされることで影響を受ける一連の行のうちの特定の行を識別します。テーブル名は、営業を受ける一連の行全体を識別します。SQL オペレーションのトリガーにより影響を受ける各行は、列を特定の相関名で分類することで、トリガーされたアクションで利用できます。PostgreSQL ではこの機能はサポートされていません。NEW または OLD 相関名のみが使用されます。
- INSTEAD OF トリガー – AWS SCT は、これらをサポートしています。

Db2 LUW パーティションテーブルから PostgreSQL バージョン 10 パーティションテーブルへの変換

AWS SCT では Db2 LUW テーブルを PostgreSQL 10 のパーティションテーブルに変換できます。Db2 LUW パーティションテーブルを PostgreSQL に変換する場合はいくつかの制限があります。

- Db2 LUW では null 値が許容される列をもつパーティションテーブルを作成したり、パーティションに null 値を保存するように指定できます。しかし、PostgreSQL は RANGE パーティションの NULL 値をサポートしていません。
- Db2 LUW は INCLUSIVE または EXCLUSIVE 句を使用して範囲境界値を設定できます。PostgreSQL は開始境界で INCLUSIVE のみ、終了境界で EXCLUSIVE のみをサポートします。変換されたパーティション名の形式は <original_table_name>_<original_partition_name> です。
- Db2 LUW のパーティションテーブルでプライマリキーや固有キーを作成できます。PostgreSQL では、プライマリキーや固有キーをパーティションごとに直接作成する必要があります。プライマ

リキーや固有キーの制約を親テーブルから削除する必要があります。変換されたキー名の形式は <original_key_name>_<original_partition _name> です。

- Db2 LUW ではパーティションテーブルから、またはテーブルに、外部キー制約を作成できます。しかし、PostgreSQL はパーティションテーブルで外部キーの参照をサポートしていません。また、PostgreSQL はパーティションテーブルから別のテーブルへの外部キーの参照をサポートしていません。
- Db2 LUW ではパーティションテーブルにインデックスを作成できます。しかし PostgreSQL では、インデックスをパーティションごとに直接作成する必要があります。親テーブルからインデックスを削除する必要があります。変換されたインデックス名の形式は <original_index_name>_<original_partition_name> です。
- 行トリガーはパーティションテーブルではなく、個別のパーティションで定義する必要があります。親テーブルからトリガーを削除する必要があります。変換されたトリガー名の形式は <original_trigger_name>_<original_partition_name> です。

PostgreSQL をターゲットとする場合の権限

PostgreSQL をターゲットとして使用するには、AWS SCT に `CREATE ON DATABASE` 権限が必要です。ターゲット PostgreSQL データベースごとにこの権限を必ず付与してください。

変換されたパブリックシノニムを使用するには、データベースのデフォルト検索パスを `"$user", public_synonyms, public` に変更します。

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

上記の例の *user_name* は使用するユーザー名に置き換えます。[*db_name*] をターゲットデータベースの名前に置き換えます。最後に、[*your_password*] を安全なパスワードに置き換えます。

PostgreSQL では、スキーマを削除できるのはスキーマ所有者または `superuser` のみです。スキーマ所有者が一部のオブジェクトを所有していない場合でも、スキーマとスキーマに含まれるすべてのオブジェクトを削除できます。

異なるユーザーを使用して異なるスキーマを変換してターゲットデータベースに適用すると、AWS SCT で、スキーマを削除できないときにエラーメッセージが表示されることがあります。このエラーメッセージを回避するには、`superuser` ロールを使用してください。

Db2 LUW から Amazon RDS for MySQL または Amazon Aurora MySQL への変換

IBM Db2 LUW データベースを RDS for MySQL または Amazon Aurora MySQL に変換するときは、次の点に注意してください。

MySQL をターゲットとする場合の権限

ターゲットとして MySQL に必要な権限を以下に示します。

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- SELECT ON mysql.proc
- INSERT, UPDATE ON AWS_DB2_EXT.*
- INSERT, UPDATE, DELETE ON AWS_DB2_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_DB2_EXT_DATA.*

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
```

```
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT SELECT ON mysql.proc TO 'user_name';
GRANT INSERT, UPDATE ON AWS_DB2_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_DB2_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_DB2_EXT_DATA.* TO 'user_name';
```

上記の例の *user_name* は使用するユーザー名に置き換えます。次に、*your_password* を安全なパスワードに置き換えます。

Amazon RDS for MySQL または Aurora MySQL をターゲットとして使用するには、lower_case_table_names パラメータを 1 に設定します。この値は、MySQL サーバーがテーブル、インデックス、トリガー、データベースなどのオブジェクト名の識別子を、大文字と小文字を区別せずに処理することを意味します。ターゲットインスタンスでバイナリログを有効にしている場合は、log_bin_trust_function_creators パラメータを 1 と設定します。この場合、ストアド関数を作成するのに、DETERMINISTIC 特性、READS SQL DATA 特性、NO SQL 特性を使用する必要はありません。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

AWS SCT ソースとしての MySQL の使用

AWS SCT を使用して、MySQL からのスキーマ、データベースコードオブジェクトおよびアプリケーションコードを次のターゲットに変換できます。

- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション
- Amazon RDS for MySQL

詳細については、次のセクションを参照してください。

トピック

- ターゲットデータベースとしての PostgreSQL の権限

ソースデータベースとしての MySQL の権限

ソースとして MySQL に必要な権限を以下に示します。

- SELECT ON *.*
- SHOW VIEW ON *.*

ソースとしての MySQL への接続

以下の手順を使用して、AWS Schema Conversion Tool を使用する MySQL ソースデータベースに接続します。

MySQL ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [MySQL] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。AWS SCT で、この名前が左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット]で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager からのデータベース認証情報の使用については、AWS Secrets Manager を使用する を参照してください。
   - MySQL ソースデータベースの接続情報を手動で入力するには、以下の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。

パラメータ	アクション
	IPv6 アドレスプロトコルを使用してソース MySQL データベースに接続できます。そのためには、次の例に示すように IP アドレスの入力に必ず角括弧を使用します。 `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソース データベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。

パラメータ	アクション
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [SSL を必須]: SSL を使用してのみサーバーに接続する場合は、このオプションを選択します。 [Require SSL] (SSL を必須) を選択した場合は、サーバーで SSL がサポートされていないと、サーバーに接続することはできません。[Require SSL] (SSL を必須) を選択しなかった場合は、サーバーで SSL がサポートされていなくても、SSL を使用せずにサーバーに接続できます。詳細については、「セキュア接続を使用するように MySQL を設定する」を参照してください。 • [サーバー証明書を確認する]: 信頼ストアを使用してサーバー証明書を確認するには、このオプションを選択します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションを有効にすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[MySql ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。

6. [Connect] (接続) を選択して、ソースデータベースに接続します。

ターゲットデータベースとしての PostgreSQL の権限

PostgreSQL をターゲットとして使用するには、AWS SCT で CREATE ON DATABASE 権限が必要です。ターゲット PostgreSQL データベースごとにこの権限を必ず付与してください。

変換されたパブリックシノニムを使用するには、データベースのデフォルト検索パスを "$user", public_synonyms, public に変更します。

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

前述の例では、[*user_name*] をお客様の設定のユーザー名に置き換えます。[*db_name*] をターゲットデータベースの名前に置き換えます。最後に、[*your_password*] を安全なパスワードに置き換えます。

PostgreSQL では、スキーマの所有者または superuser だけがスキーマを削除できます。スキーマの所有者が一部のオブジェクトを所有していなくても、所有者はスキーマとそのスキーマに含まれるすべてのオブジェクトを削除できます。

異なるユーザーを使用して異なるスキーマを変換してターゲットデータベースに適用すると、AWS SCT がスキーマを削除できないときに、エラーメッセージが表示されることがあります。このエラーメッセージを回避するには、superuser ロールを使用してください。

AWS SCT のソースとしての Oracle Database の使用

AWS SCT を使用して、Oracle Database からのスキーマ、データベースコードオブジェクトおよびアプリケーションコードを次のターゲットに変換できます。

- Amazon RDS for MySQL
- Amazon Aurora MySQL 互換エディション
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション
- Amazon RDS for Oracle

- Amazon RDS for MariaDB

ソースが Oracle データベースである場合、コメントを適切な形式 (PostgreSQL データベースなど) に変換することができます。AWS SCT は、テーブル、ビュー、および列のコメントを変換できます。コメントにはアポストロフィを含めることができます。AWS SCT は、文字列リテラルの場合と同じように、SQL ステートメントを変換するときにアポストロフィを二重にします。

詳細については、以下を参照してください。

トピック

ソースとしての Oracle の権限

ソースとして Oracle に必要な権限を以下に示します。

- CONNECT
- SELECT_CATALOG_ROLE
- SELECT ANY DICTIONARY
- SELECT ON SYS.ARGUMENT$

ソースとしての Oracle への接続

以下の手順を使用して、AWS Schema Conversion Tool を使用する Oracle ソースデータベースに接続します。

Oracle ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Oracle] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。

3. [接続名] にデータベースの名前を入力します。AWS SCT で、この名前が左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。

     1. [AWS シークレット]で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager からのデータベース認証情報の使用については、「AWS Secrets Manager を使用する」を参照してください。

   - Oracle ソースデータベースの接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション

タイプ	データベースへの接続タイプを選択します。選択したタイプに応じて、以下の追加情報を提供します。
	• SID
	  • [サーバー名]: ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
	  • Server port: ソースデータベースサーバーへの接続に使用するポート。
	  • Oracle SID: Oracle System ID (SID)。Oracle SID を見つけるには、Oracle データベースに対して以下のクエリを発行します。
	`SELECT sys_context('userenv','instance_name') AS SID FROM dual;`
	• [サービス名]
	  • Server name: ソースデータベースサーバーの DNS 名または IP アドレス。
	IPv6 アドレスプロトコルを使用してソース Oracle データベースに接続できます。そのためには、次の例

パラメータ	アクション
	に示すように IP アドレスを入力するのに角括弧を使用するようにしてください。 `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]` • Server port: ソースデータベースサーバーへの接続に使用するポート。 • [サービス名]: 接続先の Oracle サービスの名前。 • [TNS Alias] • TNS file path: Transparent Network Substrate (TNS) 名接続情報を含むファイルへのパス。 TNS ファイルを選択したら、AWS SCT がそのファイルからのすべての Oracle データベース接続を [TNS エイリアス] リストに追加します。 Oracle リアルアプリケーションクラスタ (RAC) に接続するには、このオプションを選択します。 • [TNS エイリアス]: ソースデータベースへの接続に使用するこのファイルからの TNS エイリアス。 • [TNS Connect Identifier] • [TNS 識別子]: 登録された TNS 接続情報の識別子。

パラメータ	アクション
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 Oracle データベースに初めて接続するときに、Oracle ドライバファイル (ojdbc8.jar) のパスを入力します。このファイルは http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html でダウンロードできます。ダウンロードを完了するには、無料の Oracle Technical Network ウェブサイトに登録する必要があります。AWS SCT は、その後の Oracle データベース接続で、選択されたドライバを使用します。ドライバのパスは、[グローバル設定] の [ドライバー] タブを使用して変更できます。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [SSL 認証]:証明書による SSL 認証を使用する場合はこのオプションを選択します。[設定]、[グローバル設定]、[セキュリティ] で信頼ストアとキーストアを設定します。 • [信頼ストア]: 使用する信頼ストア。 • [キーストア]: 使用するキーストア。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションを選択してデータベースのパスワードを保存し、パスワードを入力しなくてもデータベースにすばやく接続できます。

パラメータ	アクション
[Oracle ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Oracle から Amazon RDS for PostgreSQL または Amazon Aurora PostgreSQL への変換

Oracle データベースを RDS for PostgreSQL または Amazon Aurora PostgreSQL に変換するときは、次の点に注意してください。

トピック

- ターゲットデータベースとしての PostgreSQL の権限
- Oracle から PostgreSQL へのコンバージョン設定
- Oracle シーケンスの変換
- Oracle ROWID の変換
- Oracle ダイナミック SQL の変換
- Oracle パーティションの変換

Oracle システムオブジェクトを PostgreSQL に変換すると、AWS SCT で、次の表に示すように変換が実行されます。

Oracle システムオブジェクト	説明	変換された PostgreSQL オブジェクト
V$VERSION	Oracle Database のコアライブラリのコンポーネントのバージョン番号を表示します	aws_oracle_ext.v$version
V$INSTANCE	現在のインスタンスの状態を示すビュー。	aws_oracle_ext.v$instance

AWS SCT を使用して Oracle SQL*Plus ファイルを psql に変換できます。psql は PostgreSQL のターミナルベースのフロントエンドです。詳細については、「AWS SCT を使用したアプリケーション SQL の変換」を参照してください。

ターゲットデータベースとしての PostgreSQL の権限

PostgreSQL をターゲットとして使用するには、AWS SCT に `CREATE ON DATABASE` 権限が必要です。ターゲット PostgreSQL データベースごとにこの権限を必ず付与してください。

変換されたパブリックシノニムを使用するには、データベースのデフォルト検索パスを `"$user", public_synonyms, public` に変更します。

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

前述の例では、[*user_name*] をお客様の設定のユーザー名に置き換えます。[*db_name*] をターゲットデータベースの名前に置き換えます。最後に、[*your_password*] を安全なパスワードに置き換えます。

Amazon RDS for PostgreSQL AWS SCT をターゲットとして使用するには、`rds_superuser` 権限が必要です。

PostgreSQL では、スキーマの所有者または `superuser` だけがスキーマを削除できます。スキーマの所有者が一部のオブジェクトを所有していなくても、所有者はスキーマとそのスキーマに含まれるすべてのオブジェクトを削除できます。

異なるユーザーを使用して異なるスキーマを変換してターゲットデータベースに適用すると、AWS SCT がスキーマを削除できないときにエラーメッセージが表示されることがあります。このエラーメッセージを回避するには、superuser ロールを使用してください。

Oracle から PostgreSQL へのコンバージョン設定

Oracle から PostgreSQL への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Oracle] を選択し、次に [Oracle — PostgreSQL] を選択します。AWS SCT に、Oracle から PostgreSQL への変換に使用可能なすべての設定が表示されます。

AWS SCT の Oracle から PostgreSQL への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクションアイテムに関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  たとえば、変換したコード内のコメントの数を最小限に抑えるには、[エラーのみ] を選択します。変換したコードのすべてのアクション項目にコメントを含めるには、[すべてのメッセージ] を選択します。
- Oracle マテリアライズドビューを PostgreSQL AWS SCT のテーブルまたはマテリアライズドビューに変換できるようにする。[マテリアライズドビュー変換] では、ソースマテリアライズドビューを変換する方法を選択します。
- PostgreSQL がサポートしていないパラメータを持つTO_CHAR、TO_DATE、TO_NUMBER 関数が含まれているソースの Oracle コードを操作する。デフォルトでは、AWS SCT は変換後のコードでこれらのパラメータの使用法をエミュレートします。

  ソース Oracle コードに PostgreSQL がサポートするパラメータのみが含まれている場合は、ネイティブの PostgreSQL TO_CHAR、TO_DATE、TO_NUMBER 関数を使用できます。この場合、変換されたコードの処理が速くなります。これらのパラメータのみを含めるには、以下の値を選択します。
  - 関数 TO_CHAR () は Oracle 固有のフォーマット文字列を使用しません。
  - 関数 TO_DATE () は Oracle 固有のフォーマット文字列を使用しません
  - 関数 TO_NUMBER () は Oracle 固有のフォーマット文字列を使用しません
- ソース Oracle データベースが NUMBER データ型のプライマリキー列または外部キー列に整数値のみを保存している場合に対処するため、AWS SCT はこれらの列を BIGINT データ型に変換できます。この方法を実行すると、変換されたコードのパフォーマンスが向上します。この方法を採用す

るには、[NUMBER のプライマリキー列と外部キー列を BIGINT 列に変換] を選択します。データ損失を防ぐため、ソースのこれらの列には浮動小数点値が含まれないようにしてください。

- ソースコード内の非アクティブ化されたトリガーと制約をスキップする。それを行うには、[無効にされたトリガーと制約を無視する] を選択します。
- AWS SCT を使用して、動的 SQL と呼ばれる文字列変数の変換を行います。データベースコードはこれらの文字列変数の値を変更できます。AWS SCT が常にこのような文字列変数の最新の値を変換するように設定するには、[呼び出されたルーチンで作成された動的 SQL コードを変換] を選択する。
- これに対処するために、PostgreSQL バージョン 10 以前ではプロシージャがサポートされていません。PostgreSQL でのプロシージャの使用に慣れていない場合は、AWS SCT が Oracle プロシージャを PostgreSQL 関数に変換できます。そのためには、[プロシージャを関数に変換] を選択します。
- 発生したアクション項目に関する追加情報を確認するには。これを行うには、[次の重大度レベルの移行問題に対する例外発生ブロックに追加] を選択することで、拡張パックに特定の関数を追加できます。次に、ユーザー定義の例外を発生させる重要度レベルを選択します。
- 自動的に生成された名前による制約を含む可能性があるソース Oracle データベースを操作する。ソースコードでこれらの名前を使用している場合は、必ず [システムによって生成された制約名をソースの元の名前を使用して変換] を選択してください。ソースコードでこれらの制約は使われているが名前は使われていない場合は、このオプションをオフにすると変換速度が上がります。
- データベースとアプリケーションが異なるタイムゾーンで実行されているかどうかを調べるためです。デフォルトでは、AWS SCT は変換されたコードのタイムゾーンをエミュレートします。ただし、データベースとアプリケーションが同じタイムゾーンを使用している場合は、このエミュレーションは必要ありません。この場合、[クライアント側のタイムゾーンをサーバーのタイムゾーンと一致させる] を選択してください。
- ソースデータベースとターゲットデータベースが異なるタイムゾーンで動作しているかどうかを確認する。その場合、SYSDATE の Oracle 組み込み関数をエミュレートする関数は、ソース関数とは異なる値を返します。ソース関数とターゲット関数が同じ値を返すようにするには、[SYSDATE エミュレーションのデフォルトタイムゾーンを設定] を選択します。
- 変換したコードで Oracle エクステンションの関数を使用するには。そのためには、[Oracle 実装を使用する] で、使用する関数を選択します。orafce についての詳細は、GitHub で「orafce」を参照してください。

Oracle シーケンスの変換

AWS SCT はシーケンスを Oracle から PostgreSQL に変換します。シーケンスを使用して整合性制約を維持する場合は、移行したシーケンスの新しい値が既存の値と重複しないようにしてください。

変換後のシーケンスにソースデータベースの最後の値を代入するには

1. Oracle AWS SCT をソースとしてプロジェクトを開きます。
2. [設定] を選択し、[変換設定] を選択します。
3. 上のリストから [Oracle] を選択し、次に [Oracle — PostgreSQL] を選択します。AWS SCT に、Oracle から PostgreSQL への変換に使用可能なすべての設定が表示されます。
4. [変換したシーケンスにソース側で最後に生成された値を入力] を選択します。
5. [OK] を選択して設定を保存し、[Current project settings] (現在のプロジェクト設定) ダイアログボックスを閉じます。

Oracle ROWID の変換

Oracle データベースの ROWID 疑似列にはテーブルの行のアドレスが含まれます。ROWID 疑似列は Oracle に固有であるため、AWS SCT は ROWID 疑似列を PostgreSQL 上のデータ列に変換します。この変換を使用すると、ROWID 情報を保持できます。

ROWID 疑似列を変換するとき、AWS SCT はその `bigint` データ型でデータ列を作成できます。プライマリキーが存在しない場合、AWS SCT は ROWID 列をプライマリキーとして設定します。プライマリキーが存在する場合、AWS SCT は、一意の制約事項として ROWID 列を設定します。

ソースデータベースのコードに、数値データ型では実行できない ROWID の操作が含まれている場合は、AWS SCT はその `character varying` データ型でデータ列を作成できます。

Oracle ROWID のデータ列をプロジェクト用に作成するには

1. Oracle AWS SCT をソースとしてプロジェクトを開きます。
2. [設定] を選択し、[変換設定] を選択します。
3. 上のリストから [Oracle] を選択し、次に [Oracle — PostgreSQL] を選択します。AWS SCT に、Oracle から PostgreSQL への変換に使用可能なすべての設定が表示されます。
4. [行 ID を生成] では、次のいずれかを実行します。
   - [ID として生成] を選択し、数値データ列を作成します。
   - [文字ドメインタイプとして生成] を選択し、文字データ列を作成します。

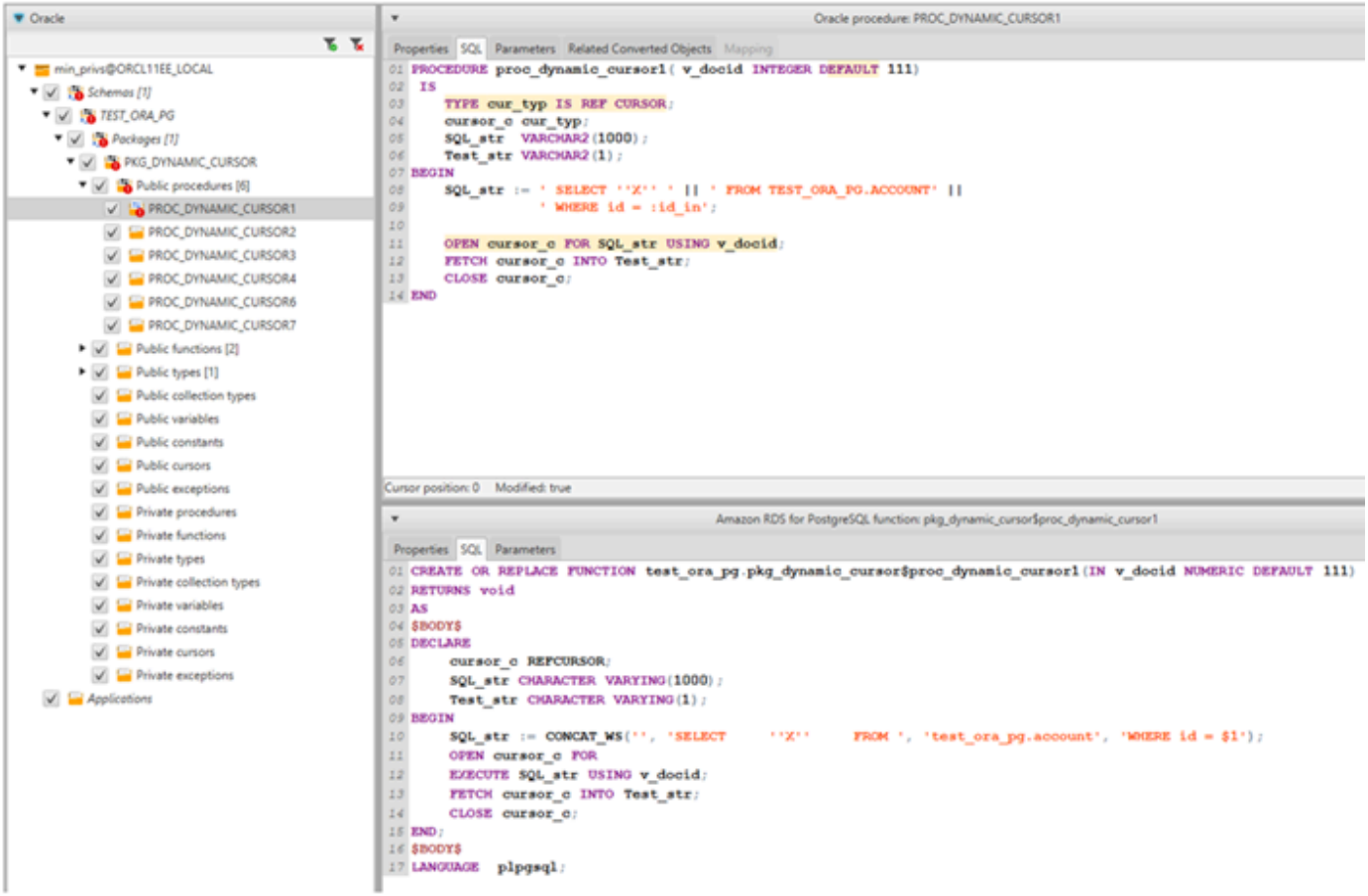
5. [OK] を選択して設定を保存し、[Current project settings] (現在のプロジェクト設定) ダイアログボックスを閉じます。

Oracle ダイナミック SQL の変換

オラクルでは動的 SQL を実装する方法として、EXECUTE IMMEDIATE ステートメントを使用する方法と DBMS_SQL パッケージ内のプロシージャを呼び出す方法の 2 つがあります。ソース Oracle データベースに動的 SQL を含むオブジェクトが含まれている場合は、AWS SCT を使用して Oracle 動的 SQL ステートメントを PostgreSQL に変換します。

Oracle の動的 SQL を PostgreSQL に変換するには

1. Oracle をソースとして AWS SCT プロジェクトを開きます。
2. Oracle ソースツリービューで動的 SQL を使用するデータベースオブジェクトを選択します。
3. オブジェクトのコンテキスト (右クリック) メニューを開き、[。スキーマの変換] を選択して、存在する場合はオブジェクトを置き換えます。次のスクリーンショットは、動的 SQL を使用した Oracle プロシージャの下に変換されたプロシージャを示しています。



Oracle パーティションの変換

AWS SCT では現在、以下のパーティションメソッドをサポートします。

- [Range] (範囲)
- リスト
- 複数列の範囲
- ハッシュ
- コンポジット (リスト - リスト、レンジ - リスト、リスト - レンジ、リスト - ハッシュ、レンジ - ハッシュ、ハッシュ - ハッシュ)

Oracle から Amazon RDS for MySQL または Amazon Aurora MySQL への変換

変換した MySQL コードで Oracle データベース関数をエミュレートするには、AWS SCT で Oracle から MySQL への拡張パックを使用します。拡張機能の詳細については、「AWS SCT 拡張パックの使用」を参照してください。

トピック

- MySQL をターゲットデータベースとする場合の権限
- Oracle から MySQL への変換設定
- 移行に関する考慮事項
- Oracle の WITH ステートメントから RDS for MySQL または Amazon Aurora MySQL への変換

MySQL をターゲットデータベースとする場合の権限

ターゲットとして MySQL に必要な権限を以下に示します。

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*

- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- CREATE TEMPORARY TABLES ON *.*
- AWS_LAMBDA_ACCESS
- INSERT, UPDATE ON AWS_ORACLE_EXT.*
- INSERT, UPDATE, DELETE ON AWS_ORACLE_EXT_DATA.*

バージョン 5.7 以前の MySQL データベースをターゲットとして使用する場合は、AWS_LAMBDA_ACCESS の代わりに INVOKE LAMBDA *.* 権限を付与します。MySQL データベースバージョン 8.0 以降の場合は、AWS_LAMBDA_ACCESS 権限を付与します。

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON *.* TO 'user_name';
GRANT AWS_LAMBDA_ACCESS TO 'user_name';
GRANT INSERT, UPDATE ON AWS_ORACLE_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_ORACLE_EXT_DATA.* TO 'user_name';
```

前述の例では、[*user_name*] をお客様の設定のユーザー名に置き換えます。*your_password* を安全なパスワードに置き換えます。

バージョン 5.7 以前の MySQL データベースをターゲットとして使用する場合は、GRANT AWS_LAMBDA_ACCESS TO '*user_name*' の代わりに GRANT INVOKE LAMBDA ON *.* TO '*user_name*' を使用してください。

Amazon RDS for MySQL または Aurora MySQL をターゲットとして使用するには、lower_case_table_names パラメータを 1 に設定します。この値は、MySQL サーバーがテーブル、インデックス、トリガー、データベースなどのオブジェクト名の識別子を、大文字と小文字を区別せずに処理することを意味します。ターゲットインスタンスでバイナリログを有効にしている場合は、log_bin_trust_function_creators パラメータを 1 と設定します。この場合、ストアドファンクションを作成するのに、DETERMINISTIC 特性、READS SQL DATA 特性、NO SQL 特性を使用する必要はありません。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

Oracle から MySQL への変換設定

Oracle から MySQL への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Oracle] を選択し、次に [Oracle — MySQL] を選択します。AWS SCT に、Oracle から MySQL への変換に使用可能なすべての設定が表示されます。

AWS SCT の Oracle から MySQL への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクションアイテムに関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  たとえば、変換したコード内のコメントの数を最小限に抑えるには、[エラーのみ] を選択します。変換したコードのすべてのアクション項目にコメントを含めるには、[すべてのメッセージ] を選択します。
- これに対処するため、ソース Oracle ROWID データベースでは疑似列を使用できますが、MySQL は同様の機能をサポートしていません。AWS SCT は変換されたコード内の ROWID 疑似列を エミュレートできます。そのためには、[行 ID を生成しますか] で [ID として生成] を選択します。

  Oracle のソースコードで ROWID 疑似列を使用していない場合は、[行 ID を生成しますか] で [生成しない] を選択します。この場合、変換されたコードの処理が速くなります。
- MySQL がサポートしていないパラメータを持つTO_CHAR、TO_DATE、TO_NUMBER 関数が含まれているソースの Oracle コードを操作する。デフォルトでは、AWS SCT は変換後のコードでこれらのパラメータの使用法をエミュレートします。

ソース Oracle コードに PostgreSQL がサポートするパラメータのみが含まれている場合は、ネイティブの MySQL TO_CHAR、TO_DATE、TO_NUMBER 関数を使用できます。この場合、変換されたコードの処理が速くなります。これらのパラメータのみを含めるには、以下の値を選択します。

- 関数 TO_CHAR () は Oracle 固有のフォーマット文字列を使用しません。
- 関数 TO_DATE () は Oracle 固有のフォーマット文字列を使用しません
- 関数 TO_NUMBER () は Oracle 固有のフォーマット文字列を使用しません

- データベースとアプリケーションが異なるタイムゾーンで実行されているかどうかを調べる。デフォルトでは、AWS SCT は変換されたコードのタイムゾーンをエミュレートします。ただし、データベースとアプリケーションが同じタイムゾーンを使用している場合は、このエミュレーションは必要ありません。この場合、[クライアント側のタイムゾーンをサーバーのタイムゾーンと一致させる] を選択してください。

移行に関する考慮事項

Oracle を RDS for MySQL または Aurora MySQL に変換する場合、ステートメントの実行順序を変更するには、GOTO ステートメントとラベルを使用できます。GOTO ステートメントに続くすべての GOTO ステートメントはスキップされ、ラベルで処理が継続されます。GOTO ステートメントとラベルは、プロシージャ、バッチ、またはステートメント ブロック内のどこでも使用できます。GOTO ステートメントをネストすることもできます。

MySQL は GOTO ステートメントを使用しません。AWS SCT が GOTO ステートメントが含まれるコードを変換する場合、ステートメントは BEGIN…END または LOOP…END LOOP ステートメントを使用するように変換されます。

次の表は、AWS SCT が GOTO ステートメントを変換する方法の一例です。

Oracle ステートメント	MySQL ステートメント
BEGIN .... statement1; .... GOTO label1; statement2; label1: Statement3;	BEGIN label1: BEGIN .... statement1; .... LEAVE label1; statement2; ....

Oracle ステートメント	MySQL ステートメント
.... END	END; Statement3; .... END
BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	BEGIN .... statement1; .... label1: LOOP statement2; .... ITERATE label1; LEAVE label1; END LOOP; statement3; .... statement4; .... END

BEGIN
....
statement1;
....
label1:
statement2;
....
statement3;
....
statement4;
....
END
	BEGIN
	....
	statement1;
	....
	label1:
	BEGIN
	statement2;
	....
	statement3;
	....
	statement4;
	....
	END;
	END

Oracle の WITH ステートメントから RDS for MySQL または Amazon Aurora MySQL への変換

名前 (query_name) をサブクエリのブロックに割り当てるには、WITH 句 (subquery_factoring) を Oracle で使用します。サブクエリブロックをクエリの複数の場所で参照するには、query_name を指定します。サブクエリブロックにリンクまたはパラメータ (ローカル、プロシージャ、関数、パッケージ) が含まれていない場合、AWS SCT はその句をビューまたは一時テーブルに変換します。

句を一時テーブルに変換する利点は、サブクエリへの繰り返し参照がより効率的であることです。各参照によって要求されるのではなく、一時テーブルからデータが簡単に取得されるために効率が向上します。追加のビューまたは一時テーブルを使用して、これをエミュレートできます。ビュー名には `<procedure_name>$<subselect_alias>` の形式が使用されます。

次の表で例を確認できます。

Oracle ステートメント	MySQL ステートメント

```
CREATE PROCEDURE
 TEST_ORA_PG.P_WITH_SELECT_V
ARIABLE_01
     (p_state IN NUMBER)
AS
  l_dept_id NUMBER := 1;
BEGIN
FOR cur IN
           (WITH dept_empl(id, name,
 surname,
              lastname, state, dept_id)
              AS
                  (
                    SELECT id, name,
 surname,
                     lastname, state,
 dept_id
                      FROM test_ora_
pg.dept_employees
                     WHERE state =
 p_state AND
                       dept_id =
 l_dept_id)
            SELECT id,state
```

```
CREATE PROCEDURE test_ora_pg.P_WITH
_SELECT_VARIABLE_01(IN par_P_STATE
 DOUBLE)
BEGIN
    DECLARE var_l_dept_id DOUBLE
 DEFAULT 1;
    DECLARE var$id VARCHAR (8000);
    DECLARE var$state VARCHAR (8000);
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT
        ID, STATE
        FROM (SELECT
            ID, NAME, SURNAME,
 LASTNAME, STATE, DEPT_ID
            FROM TEST_ORA_PG.DEPT_E
MPLOYEES
            WHERE STATE = par_p_sta
te AND DEPT_ID = var_l_dept_id) AS
 dept_empl
        ORDER BY ID;
    DECLARE CONTINUE HANDLER FOR NOT
 FOUND
        SET done := TRUE;
    OPEN cur;
```

Oracle ステートメント	MySQL ステートメント

FROM dept_empl
ORDER BY id) LOOP
NULL;
END LOOP;
	read_label:
	LOOP
	FETCH cur INTO var$id, var
	$state;
	
	IF done THEN
	LEAVE read_label;
	END IF;
	
	BEGIN
	END;
	END LOOP;
	CLOSE cur;
	END;

Oracle ステートメント

```
CREATE PROCEDURE
 TEST_ORA_PG.P_WITH_SELECT_R
EGULAR_MULT_01
AS
BEGIN

 FOR cur IN  (
               WITH dept_empl AS
                   (
                        SELECT id,
 name, surname,
                         lastname,
 state, dept_id
                          FROM
 test_ora_pg.dept_employees
                         WHERE state =
 1),
                    dept AS
                   (SELECT id deptid,
 parent_id,
                      name deptname
                      FROM test_ora_
pg.department
                   )
                SELECT dept_empl
.*,dept.*
                 FROM dept_empl, dept
                 WHERE dept_empl
.dept_id = dept.deptid
              ) LOOP
              NULL;
            END LOOP;
```

MySQL ステートメント

```
CREATE VIEW TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept_empl
 `(id, name, surname, lastname, state,
 dept_id)
AS
(SELECT id, name, surname, lastname,
 state, dept_id
   FROM test_ora_pg.dept_employees
  WHERE state = 1);

CREATE VIEW TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept
 `(deptid, parent_id,deptname)
AS
(SELECT id deptid, parent_id, name
 deptname
   FROM test_ora_pg.department);


CREATE PROCEDURE test_ora_pg.P_WITH
_SELECT_REGULAR_MULT_01()
BEGIN
    DECLARE var$ID DOUBLE;
    DECLARE var$NAME VARCHAR (30);
    DECLARE var$SURNAME VARCHAR (30);
    DECLARE var$LASTNAME VARCHAR (30);
    DECLARE var$STATE DOUBLE;
    DECLARE var$DEPT_ID DOUBLE;
    DECLARE var$deptid DOUBLE;
    DECLARE var$PARENT_ID DOUBLE;
    DECLARE var$deptname VARCHAR
 (200);
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT
        dept_empl.*, dept.*
        FROM TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept_empl
          ` AS dept_empl,
            TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept
          ` AS dept
```

Oracle ステートメント	MySQL ステートメント

```
        WHERE dept_empl.DEPT_ID =
 dept.DEPTID;
    DECLARE CONTINUE HANDLER FOR NOT
 FOUND
        SET done := TRUE;
    OPEN cur;

    read_label:
    LOOP
    FETCH cur INTO var$ID, var$NAME,
 var$SURNAME,
     var$LASTNAME, var$STATE, var
$DEPT_ID, var$deptid,
     var$PARENT_ID, var$deptname;

        IF done THEN
            LEAVE read_label;
        END IF;

        BEGIN
        END;
    END LOOP;
    CLOSE cur;
END;

call test_ora_pg.P_WITH_SELECT_R
EGULAR_MULT_01()
```

Oracle ステートメント	MySQL ステートメント

```
CREATE PROCEDURE
  TEST_ORA_PG.P_WITH_SELECT_V
AR_CROSS_02(p_state IN NUMBER)
AS
   l_dept_id NUMBER := 10;
BEGIN
 FOR cur IN  (
               WITH emp AS

                    (SELECT id, name,
 surname,
                      lastname, state,
 dept_id
                       FROM test_ora_
pg.dept_employees
                      WHERE dept_id >
 10
                    ),
                    active_emp AS
                    (
                      SELECT id
                        FROM emp
                       WHERE emp.state
 = p_state
                    )

                SELECT *
                  FROM active_emp

             ) LOOP
        NULL;
  END LOOP;

END;
```

```
CREATE VIEW TEST_ORA_PG.`P_WIT
H_SELECT_VAR_CROSS_01$emp
    `(id, name, surname, lastname,
 state, dept_id)
AS
(SELECT
       id, name, surname, lastname,
       state, dept_id
  FROM TEST_ORA_PG.DEPT_EMPLOYEES
  WHERE DEPT_ID > 10);


CREATE PROCEDURE
   test_ora_pg.P_WITH_SELECT_V
AR_CROSS_02(IN par_P_STATE DOUBLE)
BEGIN
    DECLARE var_l_dept_id DOUBLE
 DEFAULT 10;
    DECLARE var$ID DOUBLE;
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT *
                               FROM
 (SELECT

 ID

 FROM
                               TEST_ORA_
PG.
                                `P_WITH_S
ELECT_VAR_CROSS_01$emp` AS emp

 WHERE emp.STATE = par_p_state)
                                     AS
 active_emp;
    DECLARE CONTINUE HANDLER FOR NOT
 FOUND
        SET done := TRUE;
    OPEN cur;

    read_label:
```

Oracle ステートメント	MySQL ステートメント

	LOOP
	FETCH cur INTO var$ID;
	
	IF done THEN
	LEAVE read_label;
	END IF;
	
	BEGIN
	END;
	END LOOP;
	CLOSE cur;
	END;

Oracle の Amazon RDS for Oracle への変換

Oracle スキーマとコードを Amazon RDS for Oracle に移行する際は以下の点を考慮してください。

- AWS SCT ではオブジェクトツリーにディレクトリオブジェクトを追加できます。ディレクトリオブジェクトは、それぞれがサーバーのファイルシステムの物理ディレクトリを表す論理構造です。ディレクトリオブジェクトは、DBMS_LOB、UTL_FILE、DBMS_FILE_TRANSFER、DATAPUMP ユーティリティなどのパッケージで使用できます。
- AWS SCT では、Oracle テーブルスペースの Amazon RDS for Oracle DB インスタンスへの変換がサポートされています。Oracle は、論理的にはテーブルスペースに、物理的には対応するテーブルスペースに関連付けられたデータファイルに、データを保存します。Oracle では、テーブルスペースとデータファイル名を作成できます。Amazon RDS は、データファイル、ログファイル、制御ファイルとして Oracle Managed Files (OMF) のみをサポートしています。AWS SCT は変換中に必要なデータファイルを作成します。
- AWS SCT はサーバーレベルのロールと権限を変換できます。Oracle データベースエンジンではロールベースのセキュリティを使用します。ロールとは、ユーザーに対して付与または取り消すことができる権限のコレクションです。Amazon RDS の事前に定義されたロールは DBA といい、通常、Oracle データベースエンジンに対するすべての管理権限が許可されています。次の権限は、Oracle エンジンを利用する Amazon RDS DB インスタンスの DBA ロールでは使用できません。
  - データベースの変更
  - システムの変更

- ディレクトリの作成
- 権限の付与
- ロールの付与
- 外部ジョブの作成

その他の権限 (例: 高度なフィルタリングや列に関する権限) をすべて Amazon RDS for Oracle ユーザーロールに付与することができます。

- AWS SCT では、Oracle ジョブから Amazon RDS for Oracle で実行できるジョブへの変換がサポートされています。この変換には、次のモノを含むいくつかの制限があります。
  - 実行ジョブはサポートされていません。
  - ANYDATA データ型を引数として使用するスケジュールジョブはサポートされていません。
- Oracle Real Application Clusters (RAC) One Node は、Oracle Database 11g Release 2 とともに導入された Oracle Database Enterprise Edition のオプションです。Amazon RDS for Oracle は RAC 機能をサポートしていません。可用性を高めるには Amazon RDS マルチ AZ を使用してください。

  Amazon RDS のマルチ AZ 配置では、異なるアベイラビリティーゾーンに同期スタンバイレプリカが自動的にプロビジョニングされて維持されます。プライマリ DB インスタンスは、同期的にアベイラビリティーゾーン間でスタンバイレプリカにレプリケートされます。この機能により、データの冗長性が確保されて I/O のフリーズがなくなり、システムバックアップ時のレイテンシー急上昇が最小限に抑えられます。
- Oracle Spatial は、Oracle データベースのストレージ、取得、更新、および spatial データコレクションのクエリを可能にする SQL スキーマと機能を提供します。Oracle Locator は、インターネットとワイヤレスベースのアプリケーションをサポートするために一般的に必要な機能とパートナーベースの GIS ソリューションを提供します。Oracle Locator は Oracle Spatial の制限付きのサブネットです。

  Oracle Spatial 機能や Oracle Locator 機能を使用するには、SPATIAL オプションまたは LOCATOR オプション (相互に排他的) を DB インスタンスのオプショングループに追加します。

  Amazon RDS for Oracle DB インスタンスで Oracle Spatial および Oracle Locator を使用するには、いくつかの前提条件を満たす必要があります。
  - インスタンスで Oracle Enterprise Edition バージョン 12.1.0.2.v6 以降、または 11.2.0.4.v10 以降が実行されていること。
  - インスタンスが Virtual Private Cloud (VPC) 内にあること。

  - インスタンスの DB インスタンスクラスが Oracle 機能をサポート可能であること。例えば、Oracle Spatial は、db.m1.small、db.t1.micro、db.t2.micro、または db.t2.small の DB インスタンスクラスではサポートされません。詳細については、「DB インスタンスクラスの Oracle のサポート」を参照してください。
  - インスタンスで、マイナーバージョン自動アップグレードが有効化されていること。CVSS スコアが 9 以上、またはその他のセキュリティの脆弱性が報告されている場合には、Amazon RDS によって DB インスタンスが最新の Oracle PSU に更新されます。詳細については、次を参照してください。

    Oracle DB インスタンスの設定
  - DB インスタンスが バージョン 11.2.0.4.v10 以降の場合には、XMLDB オプションをインストールする必要があります。詳細については、次を参照してください。

    Oracle XML DB。
  - Oracle の Oracle Spatial ライセンスが付与されていること。詳細については、Oracle ドキュメントの Oracle Spatial and Graph を参照してください。
- Data Guard は、Oracle Database Enterprise Edition に含まれています。可用性を高めるには Amazon RDS マルチ AZ 機能を使用してください。

  Amazon RDS のマルチ AZ 配置では、異なるアベイラビリティーゾーンに同期スタンバイレプリカが自動的にプロビジョニングされて維持されます。プライマリ DB インスタンスは、同期的にアベイラビリティーゾーン間でスタンバイレプリカにレプリケートされます。この機能により、データの冗長性が確保されて I/O のフリーズがなくなり、システムバックアップ時のレイテンシー急上昇が最小限に抑えられます。
- AWS SCT では、Amazon RDS for Oracle への移行時の Oracle DBMS_SCHEDULER オブジェクトの変換がサポートされています。AWS SCT 評価レポートは、スケジュールオブジェクトが変換できるかどうかを示します。Amazon RDS でのスケジュールオブジェクトの使用の詳細については、Amazon RDS のドキュメントを参照してください。
- Oracle から Amazon RDS for Oracle の変換では、DB リンクがサポートされます。データベースリンクは、別のデータベースのオブジェクトにアクセスできるようにする、1 つのデータベース内のスキーマオブジェクトです。もう 1 つのデータベースは Oracle データベースである必要はありません。ただし、Oracle 以外のデータベースにアクセスするには、Oracle Heterogeneous Services を使用する必要があります。

  データベースリンクを作成したら、そのリンクを SQL ステートメントで使用して、他のデータベースのテーブル、ビュー、および PL/SQL オブジェクトを参照できます。データベースリンク

を使用するには、テーブル、ビュー、または PL/SQL オブジェクトの名前に @dblink を追加します。他のデータベースのテーブルまたはビューには、SELECT ステートメントを使用してクエリを実行できます。Oracle データベースリンクの使用の詳細については、Oracle のドキュメントを参照してください。

Amazon RDS でのデータベースリンクの使用の詳細については、Amazon RDS のドキュメントを参照してください。

- AWS SCT 評価レポートには、変換に関するサーバーのメトリクスが提供されています。Oracle インスタンスに関するこれらのメトリクスには、次のようなものがあります。
  - ターゲット DB インスタンスのコンピューティング能力とメモリ容量。
  - サポートされていない Oracle 機能や、Amazon RDS でサポートされていない Real Application Clusters。
  - ディスク読み取り/書き込み負荷
  - ディスク合計の平均スループット
  - サーバー情報 (例: サーバー名、OS、ホスト名、文字セット)。

ターゲットとして RDS for Oracle を使用する場合の権限

Amazon RDS for Oracle に移行するには、権限のあるデータベースユーザーを作成します。次のコード例を使用できます。

```
CREATE USER user_name IDENTIFIED BY your_password;

-- System privileges
GRANT DROP ANY CUBE BUILD PROCESS TO user_name;
GRANT ALTER ANY CUBE TO user_name;
GRANT CREATE ANY CUBE DIMENSION TO user_name;
GRANT CREATE ANY ASSEMBLY TO user_name;
GRANT ALTER ANY RULE TO user_name;
GRANT SELECT ANY DICTIONARY TO user_name;
GRANT ALTER ANY DIMENSION TO user_name;
GRANT CREATE ANY DIMENSION TO user_name;
GRANT ALTER ANY TYPE TO user_name;
GRANT DROP ANY TRIGGER TO user_name;
GRANT CREATE ANY VIEW TO user_name;
GRANT ALTER ANY CUBE BUILD PROCESS TO user_name;
GRANT CREATE ANY CREDENTIAL TO user_name;
GRANT DROP ANY CUBE DIMENSION TO user_name;
GRANT DROP ANY ASSEMBLY TO user_name;
```

```
GRANT DROP ANY PROCEDURE TO user_name;
GRANT ALTER ANY PROCEDURE TO user_name;
GRANT ALTER ANY SQL TRANSLATION PROFILE TO user_name;
GRANT DROP ANY MEASURE FOLDER TO user_name;
GRANT CREATE ANY MEASURE FOLDER TO user_name;
GRANT DROP ANY CUBE TO user_name;
GRANT DROP ANY MINING MODEL TO user_name;
GRANT CREATE ANY MINING MODEL TO user_name;
GRANT DROP ANY EDITION TO user_name;
GRANT CREATE ANY EVALUATION CONTEXT TO user_name;
GRANT DROP ANY DIMENSION TO user_name;
GRANT ALTER ANY INDEXTYPE TO user_name;
GRANT DROP ANY TYPE TO user_name;
GRANT CREATE ANY PROCEDURE TO user_name;
GRANT CREATE ANY SQL TRANSLATION PROFILE TO user_name;
GRANT CREATE ANY CUBE TO user_name;
GRANT COMMENT ANY MINING MODEL TO user_name;
GRANT ALTER ANY MINING MODEL TO user_name;
GRANT DROP ANY SQL PROFILE TO user_name;
GRANT CREATE ANY JOB TO user_name;
GRANT DROP ANY EVALUATION CONTEXT TO user_name;
GRANT ALTER ANY EVALUATION CONTEXT TO user_name;
GRANT CREATE ANY INDEXTYPE TO user_name;
GRANT CREATE ANY OPERATOR TO user_name;
GRANT CREATE ANY TRIGGER TO user_name;
GRANT DROP ANY ROLE TO user_name;
GRANT DROP ANY SEQUENCE TO user_name;
GRANT DROP ANY CLUSTER TO user_name;
GRANT DROP ANY SQL TRANSLATION PROFILE TO user_name;
GRANT ALTER ANY ASSEMBLY TO user_name;
GRANT CREATE ANY RULE SET TO user_name;
GRANT ALTER ANY OUTLINE TO user_name;
GRANT UNDER ANY TYPE TO user_name;
GRANT CREATE ANY TYPE TO user_name;
GRANT DROP ANY MATERIALIZED VIEW TO user_name;
GRANT ALTER ANY ROLE TO user_name;
GRANT DROP ANY VIEW TO user_name;
GRANT ALTER ANY INDEX TO user_name;
GRANT COMMENT ANY TABLE TO user_name;
GRANT CREATE ANY TABLE TO user_name;
GRANT CREATE USER TO user_name;
GRANT DROP ANY RULE SET TO user_name;
GRANT CREATE ANY CONTEXT TO user_name;
GRANT DROP ANY INDEXTYPE TO user_name;
```

```
GRANT ALTER ANY OPERATOR TO user_name;
GRANT CREATE ANY MATERIALIZED VIEW TO user_name;
GRANT ALTER ANY SEQUENCE TO user_name;
GRANT DROP ANY SYNONYM TO user_name;
GRANT CREATE ANY SYNONYM TO user_name;
GRANT DROP USER TO user_name;
GRANT ALTER ANY MEASURE FOLDER TO user_name;
GRANT ALTER ANY EDITION TO user_name;
GRANT DROP ANY RULE TO user_name;
GRANT CREATE ANY RULE TO user_name;
GRANT ALTER ANY RULE SET TO user_name;
GRANT CREATE ANY OUTLINE TO user_name;
GRANT UNDER ANY TABLE TO user_name;
GRANT UNDER ANY VIEW TO user_name;
GRANT DROP ANY DIRECTORY TO user_name;
GRANT ALTER ANY CLUSTER TO user_name;
GRANT CREATE ANY CLUSTER TO user_name;
GRANT ALTER ANY TABLE TO user_name;
GRANT CREATE ANY CUBE BUILD PROCESS TO user_name;
GRANT ALTER ANY CUBE DIMENSION TO user_name;
GRANT CREATE ANY EDITION TO user_name;
GRANT CREATE ANY SQL PROFILE TO user_name;
GRANT ALTER ANY SQL PROFILE TO user_name;
GRANT DROP ANY OUTLINE TO user_name;
GRANT DROP ANY CONTEXT TO user_name;
GRANT DROP ANY OPERATOR TO user_name;
GRANT DROP ANY LIBRARY TO user_name;
GRANT ALTER ANY LIBRARY TO user_name;
GRANT CREATE ANY LIBRARY TO user_name;
GRANT ALTER ANY MATERIALIZED VIEW TO user_name;
GRANT ALTER ANY TRIGGER TO user_name;
GRANT CREATE ANY SEQUENCE TO user_name;
GRANT DROP ANY INDEX TO user_name;
GRANT CREATE ANY INDEX TO user_name;
GRANT DROP ANY TABLE TO user_name;
GRANT SELECT_CATALOG_ROLE TO user_name;
GRANT SELECT ANY SEQUENCE TO user_name;

-- Database Links
GRANT CREATE DATABASE LINK TO user_name;
GRANT CREATE PUBLIC DATABASE LINK TO user_name;
GRANT DROP PUBLIC DATABASE LINK TO user_name;
```

```
-- Server Level Objects (directory)
GRANT CREATE ANY DIRECTORY TO user_name;
GRANT DROP ANY DIRECTORY TO user_name;
-- (for RDS only)
GRANT EXECUTE ON RDSADMIN.RDSADMIN_UTIL TO user_name;

-- Server Level Objects (tablespace)
GRANT CREATE TABLESPACE TO user_name;
GRANT DROP TABLESPACE TO user_name;

-- Server Level Objects (user roles)
/* (grant source privileges with admin option or convert roles/privs as DBA) */

-- Queues
grant execute on DBMS_AQADM to user_name;
grant aq_administrator_role to user_name;

-- for Materialized View Logs creation
GRANT SELECT ANY TABLE TO user_name;

-- Roles
GRANT RESOURCE TO user_name;
GRANT CONNECT TO user_name;
```

前述の例では、[*user_name*] をお客様の設定のユーザー名に置き換えます。*your_password* を安全なパスワードに置き換えます。

Oracle から Amazon RDS for Oracle に変換する際の制限

Oracle スキーマとコードを Amazon RDS for Oracle に移行する際に考慮する必要のある制限の一部を以下に示します。

- Amazon RDS の事前に定義されたロールは DBA といい、通常、Oracle データベースエンジンに対するすべての管理権限が許可されています。次の権限は、Oracle エンジンを利用する Amazon RDS DB インスタンスの DBA ロールでは使用できません。
  - データベースの変更
  - システムの変更
  - ディレクトリの作成
  - 権限の付与
  - ロールの付与

- 外部ジョブの作成

  Oracle RDS ユーザーロールに他のすべての権限を付与できます。

- Amazon RDS for Oracle では、従来の監査、DBMS_FGA パッケージを使用した詳細な監査、および Oracle 統合監査がサポートされています。
- Amazon RDS for Oracle は、変更データキャプチャ (CDC) をサポートしていません。データベースの移行中または移行後に CDC を行うには、AWS Database Migration Service を使用します。

AWS SCT のソースとしての PostgreSQL の使用

AWS SCT を使用して、PostgreSQL からのスキーマ、データベースコードオブジェクトおよびアプリケーションコードを次のターゲットに変換できます。

- Amazon RDS for MySQL
- Amazon Aurora MySQL 互換エディション
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション

詳細については、次のセクションを参照してください。

トピック

ソースデータベースとしての PostgreSQL の権限

ソースとして PostgreSQL に必要な権限を以下に示します。

- CONNECT ON DATABASE *<database_name>*
- USAGE ON SCHEMA *<database_name>*
- SELECT ON ALL TABLES IN SCHEMA *<database_name>*
- SELECT ON ALL SEQUENCES IN SCHEMA *<database_name>*

ソースとしての PostgreSQL への接続

以下の手順を使用して、AWS Schema Conversion Tool を使用する PostgreSQL ソースデータベースに接続します。

PostgreSQL ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [PostgreSQL] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] を入力します。AWS SCT の左パネルのツリーにこの名前が表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] の場合は、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - PostgreSQL ソースデータベースの接続情報を手動で入力するには、以下の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。 ソース PostgreSQL データベースには IPv6 アドレスプロトコルを使用して接続できます。そのためには、次の例に示すように IP アドレスの入力に角括弧を使用することを確認します。 `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。

パラメータ	アクション
データベース	PostgreSQL データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー認証情報を確認する]: 信頼ストアを使用してサーバー証明書を確認するには、このオプションを選択します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。この場所を [グローバル設定] セクションに表示するには、必ず追加してください。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションを有効にすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。

パラメータ	アクション
[PostgreSQL ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

MySQL をターゲットデータベースとする場合の権限

PostgreSQL から移行する場合にターゲットとして MySQL に必要な権限は次のとおりです。

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- INSERT, UPDATE ON AWS_POSTGRESQL_EXT.*
- INSERT, UPDATE, DELETE ON AWS_POSTGRESQL_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_POSTGRESQL_EXT_DATA.*

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_POSTGRESQL_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_POSTGRESQL_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_POSTGRESQL_EXT_DATA.* TO 'user_name';
```

上記の例の *user_name* は使用するユーザー名に置き換えます。次に、*your_password* を安全なパスワードに置き換えます。

Amazon RDS for MySQL または Aurora MySQL をターゲットとして使用するには、lower_case_table_names パラメータを 1 と設定します。この値は、MySQL サーバーがテーブル、インデックス、トリガー、データベースなどのオブジェクト名の識別子を、大文字と小文字を区別せずに処理することを意味します。ターゲットインスタンスでバイナリログを有効にしている場合は、log_bin_trust_function_creators パラメータを 1 と設定します。この場合、ストアドファンクションを作成するのに、DETERMINISTIC 特性、READS SQL DATA 特性、NO SQL 特性を使用する必要はありません。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

AWS SCT のソースとしての SAP ASE (Sybase ASE) の使用

AWS SCT を使用して、SAP (Sybase) Adaptive Server Enterprise (ASE) からのスキーマ、データベースコードオブジェクトおよびアプリケーションコードを次のターゲットに変換できます。

- Amazon RDS for MySQL
- Amazon Aurora MySQL 互換エディション
- Amazon RDS for MariaDB

- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション

詳細については、次のセクションを参照してください。

トピック

ソースデータベースとしての SAP ASE の特権

SAP ASE データベースをソースとして使用するには、データベースユーザーを作成して特権を付与します。これを行うには、以下のステップに従います。

データベースユーザーを作成して設定する

1. ソースデータベースに接続する。
2. 次のコマンドを使用してデータベースユーザーを作成する。新しいユーザーのパスワードを入力する。

```
USE master
CREATE LOGIN min_privs WITH PASSWORD <password>
sp_adduser min_privs
grant select on dbo.spt_values to min_privs
grant select on asehostname to min_privs
```

3. 移行するすべてのデータベースに対して、次の権限を付与する。

```
USE <database_name>
sp_adduser min_privs
grant select on dbo.sysusers to min_privs
grant select on dbo.sysobjects to min_privs
grant select on dbo.sysindexes to min_privs
```

```
grant select on dbo.syscolumns to min_privs
grant select on dbo.sysreferences to min_privs
grant select on dbo.syscomments to min_privs
grant select on dbo.syspartitions to min_privs
grant select on dbo.syspartitionkeys to min_privs
grant select on dbo.sysconstraints to min_privs
grant select on dbo.systypes to min_privs
grant select on dbo.sysqueryplans to min_privs
```

ソースとしての SAP ASE (Sybase) への接続

以下の手順を使用して、SAP ASE ソースデータベースを AWS Schema Conversion Tool に接続します。

SAP ASE ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [SAP ASE] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。

     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - SAP ASE ソースデータベースの接続情報を手動で入力するには、次の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。

パラメータ	アクション
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
データベース	SAP ASE データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 **Note** AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー認証情報を確認する]: 信頼ストアを使用してサーバー証明書を確認するには、このオプションを選択します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションを有効にすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。

パラメータ	アクション
[SAP ASE ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

MySQL をターゲットデータベースとする場合の権限

ターゲットとして MySQL に必要な権限を以下に示します。

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- INSERT, UPDATE ON AWS_SAPASE_EXT.*
- CREATE TEMPORARY TABLES ON AWS_SAPASE_EXT.*

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_SAPASE_EXT.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_SAPASE_EXT.* TO 'user_name';
```

上記の例の *user_name* は使用するユーザー名に置き換えます。次に、*your_password* を安全なパスワードに置き換えます。

Amazon RDS for MySQL または Aurora MySQL をターゲットとして使用するには、lower_case_table_names パラメータを 1 と設定します。この値は、MySQL サーバーがテーブル、インデックス、トリガー、データベースなどのオブジェクト名の識別子を、大文字と小文字を区別せずに処理することを意味します。ターゲットインスタンスでバイナリログを有効にしている場合は、log_bin_trust_function_creators パラメータを 1 と設定します。この場合、ストアドファンクションを作成するのに、DETERMINISTIC 特性、READS SQL DATA 特性、NO SQL 特性を使用する必要はありません。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

SAP ASE から MySQL への変換設定

SAP ASE から MySQL への変換設定を編集するには、[設定] を選択し、[変換設定] を選択します。上のリストから [SAP ASE] を選択し、次に [SAP ASE – MySQL] または [SAP ASE – Amazon Aurora (MySQL 互換)] を選択します。AWS SCT に、SAP ASE から PostgreSQL への変換に使用できるすべての設定が表示されます。

AWS SCT での SAP ASE から MySQL への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

[変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。

- 変換後のコードでソースデータベースオブジェクトの正確な名前を使用する。

  デフォルトでは、データベースオブジェクト、変数、AWS SCT パラメータの名前を小文字に変換します。これらの名前を元の大文字と小文字を区別しないでおくには、[ソースデータベースオブジェクト名の大文字と小文字を区別する] を選択します。ソース SAP ASE データベースサーバーで大文字と小文字を区別するオブジェクト名を使用する場合は、このオプションを選択してください。

ターゲットデータベースとしての PostgreSQL の権限

PostgreSQL をターゲットとして使用するには、AWS SCT に `CREATE ON DATABASE` 権限が必要です。ターゲット PostgreSQL データベースごとにこの権限を必ず付与してください。

変換されたパブリックシノニムを使用するには、データベースのデフォルト検索パスを `"$user", public_synonyms, public` に変更します。

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

上記の例の *user_name* は使用するユーザー名に置き換えます。[*db_name*] をターゲットデータベースの名前に置き換えます。最後に、[*your_password*] を安全なパスワードに置き換えます。

PostgreSQL では、スキーマを削除できるのはスキーマ所有者または `superuser` のみです。スキーマ所有者が一部のオブジェクトを所有していない場合でも、スキーマとスキーマに含まれるすべてのオブジェクトを削除できます。

異なるユーザーを使用して異なるスキーマを変換してターゲットデータベースに適用すると、AWS SCT がスキーマを削除できないときにエラーメッセージが表示されることがあります。このエラーメッセージを回避するには、`superuser` ロールを使用します。

SAP ASE から PostgreSQL への変換設定

SAP ASE から PostgreSQL への変換設定を編集するには、[設定] を選択し、次に [変換設定] を選択します。上のリストから [SAP ASE] を選択し、次に [SAP ASE – PostgreSQL] または [SAP ASE – Amazon Aurora (PostgreSQL 互換)] を選択します。AWS SCT に、SAP ASE から PostgreSQL への変換に使用できるすべての設定が表示されます。

AWS SCT の SAP ASE から PostgreSQL への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- 変換されるコード内のスキーマ名に使用するテンプレートを定義する。[スキーマ名生成テンプレート] では、次のオプションのいずれかを選択します。
  - [<source_db>] – PostgreSQL のスキーマ名として SAP ASE データベース名を使用します。
  - [<source_schema>] – PostgreSQL のスキーマ名として SAP ASE スキーマ名を使用します。
  - [<source_db>_<schema>] – SAP ASE データベースとスキーマ名の組み合わせを PostgreSQL のスキーマ名として使用します。
- 変換後のコードでソースデータベースオブジェクトの正確な名前を使用する。

  デフォルトでは、データベースオブジェクト、変数、AWS SCT パラメータの名前を小文字に変換します。これらの名前を元の大文字と小文字を区別しないでおくには、[ソースデータベースオブジェクト名の大文字と小文字を区別する] を選択します。ソース SAP ASE データベースサーバーで大文字と小文字を区別するオブジェクト名を使用する場合は、このオプションを選択してください。

大文字と小文字を区別する操作では、AWS SCT でデータベースオブジェクト名を小文字に変換する必要がありません。そのためには、[大文字と小文字を区別する操作では小文字へのキャストを避ける] を選択します。

- SAP ASE の異なるテーブルで同じ名前のインデックスを使用できるようにする。

  PostgreSQL では、スキーマで使用するインデックス名はすべて一意でなければなりません。AWS SCT ですべてのインデックスに固有の名前が生成されるようにするには、[インデックスの固有の名前を生成] を選択します。

Microsoft SQL Server の AWS SCT のソースとしての使用

AWS SCT を使用して、SQL Server からのスキーマ、データベースコードオブジェクトおよびアプリケーションコードを次のターゲットに変換できます。

- Amazon RDS for MySQL
- Amazon Aurora MySQL 互換エディション
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 互換エディション
- Amazon RDS for SQL Server
- Amazon RDS for MariaDB

> **Note**
>
> AWS SCT では、Amazon RDS for SQL server をソースとして使用することはサポートされていません。

以下で説明するように、AWS SCT を使用して、SQL Server から Babelfish for Aurora PostgreSQL へのスキーマ、データベースコードオブジェクト、およびアプリケーションコードの移行に関する評価レポートを作成できます。

トピック

- Microsoft SQL Server をソースとする場合の権限
- Microsoft SQL Server をソースとして使用するときの Windows 認証の使用

- ソースとしての SQL Server への接続
- SQL Server から MySQL への変換
- SQL Server から PostgreSQL への変換
- SQL Server から Amazon RDS for SQL Server への変換

Microsoft SQL Server をソースとする場合の権限

ソースとして Microsoft SQL Server に必要な権限を以下に示します。

- VIEW DEFINITION
- VIEW DATABASE STATE

`VIEW DEFINITION` 権限により、パブリックアクセスを持つユーザーはオブジェクト定義を表示できるようになります。AWS SCT は、`VIEW DATABASE STATE` 権限を使用して SQL Server Enterprise エディションの機能を確認します。

スキーマを変換する、各データベースの付与を繰り返します。

さらに、`master` データベースに次の権限を付与します。

- VIEW SERVER STATE
- VIEW ANY DEFINITION

AWS SCT は、`VIEW SERVER STATE` 権限を使用してサーバーの設定と構成を収集します。エンドポイントを表示する `VIEW ANY DEFINITION` 権限を必ず付与してください。

Microsoft Analysis Services に関する情報を読み取るには、`master` データベースで次のコマンドを実行します。

```
EXEC master..sp_addsrvrolemember @loginame = N'<user_name>', @rolename = N'sysadmin'
```

前述の例では、*<user_name>* プレースホルダーを、以前に権限を付与したユーザーの名前に置き換えます。

SQL Server エージェントに関する情報を読むには、ユーザーを `SQLAgentUser` ロールに追加します。`msdb` データベースで次のコマンドを実行します。

```
EXEC sp_addrolemember <SQLAgentRole>, <user_name>;
```

上記の例の *<SQLAgentRole>* プレースホルダーを、SQL Server エージェントのロール名に置き換えます。次に、*<user_name>* プレースホルダーを以前に権限を付与したユーザーの名前に置き換えます。詳細については、『Amazon RDS ユーザーガイド』の「SQLAgentUser ロールへのユーザーの追加」を参照してください。

ログシッピングを検出するには、msdb データベースに対する SELECT on dbo.log_shipping_primary_databases 権限を付与します。

DDL レプリケーションの通知方法を使用するには、ソースデータベースに RECEIVE ON *<schema_name>*.*<queue_name>* 権限を付与します。この例では、*<schema_name>* プレースホルダーをデータベースのスキーマ名に置き換えます。次に、*<queue_name>* プレースホルダーをキューテーブルの名前に置き換えます。

Microsoft SQL Server をソースとして使用するときの Windows 認証の使用

アプリケーションが Windows ベースのイントラネットで実行されている場合、データベースアクセスに Windows 認証を使用できることがあります。Windows 認証では、オペレーティングシステムのスレッドで確立された最新の Windows ID を使用して SQL Server データベースにアクセスします。次に、Windows ID を SQL Server データベースとアクセス権限にマッピングできます。Windows 認証を使用して SQL Server に接続するには、アプリケーションで使用している Windows ID を指定する必要があります。SQL Server データベースへのアクセス権を Windows ID に付与する必要もあります。

SQL Server には、Windows 認証モードと混合モードの 2 つのアクセスモードがあります。Windows 認証モードでは Windows 認証が有効になり、SQL Server 認証が無効になります。混合モードでは、Windows 認証と SQL Server 認証の両方が有効になります。Windows 認証は常に利用可能であり、無効にすることはできません。Windows 認証の詳細については、Microsoft Windows のドキュメントを参照してください。

TEST_DB でユーザーを作成する例を次に示します。

```
USE [TEST_DB]
CREATE USER [TestUser] FOR LOGIN [TestDomain\TestUser]
GRANT VIEW DEFINITION TO [TestUser]
GRANT VIEW DATABASE STATE TO [TestUser]
```

JDBC 接続による Windows 認証の使用

JDBC ドライバは、Windows 以外のオペレーティングシステムで使用する場合、Windows 認証をサポートしていません。ユーザー名やパスワードなどの Windows 認証情報は、Windows 以外のオペレーティングシステムから SQL Server に接続するときには自動的に指定されません。このような場合、アプリケーションは代わりに SQL Server 認証を使用する必要があります。

JDBC 接続文字列では、Windows 認証を使用して接続するには、パラメータ `integratedSecurity` を指定する必要があります。JDBC ドライバは、`integratedSecurity` 接続文字列パラメータを通じて、Windows オペレーティングシステムで統合 Windows 認証をサポートします。

認証の統合を使用するには

1. JDBC ドライバをインストールします。
2. JDBC ドライバがインストールされているコンピュータの Windows システムパス上のディレクトリに、`sqljdbc_auth.dll` ファイルをコピーします。

   `sqljdbc_auth.dll` ファイルは次の場所にインストールされます。

   <インストールディレクトリ>\sqljdbc_<バージョン>\<言語>\auth\

Windows 認証を使用して SQL Server データベースへの接続を確立しようとする場合に、「このドライバは認証の統合用に設定されていません」という工ラーが発生することがあります。この問題は、次のアクションを実行することで解決できます。

- JDBC のインストール先パスを指す 2 つの変数を宣言します。

  `variable name: SQLJDBC_HOME; variable value: D:\lib\JDBC4.1\enu`(sqljdbc4.jar が存在する場所);

  `variable name: SQLJDBC_AUTH_HOME; variable value: D\lib\JDBC4.1\enu\auth\x86`(32 ビット OS を実行している場合) または `D\lib\JDBC4.1\enu\auth\x64` (64 ビット OS を実行している場合)。これは `sqljdbc_auth.dll` の場所です。
- `sqljdbc_auth.dll` を、JDK/JRE が実行されているフォルダにコピーします。lib フォルダ、bin フォルダなどにコピーできます。例えば、次のフォルダにコピーできます。

```
[JDK_INSTALLED_PATH]\bin;
[JDK_INSTALLED_PATH]\jre\bin;
```

```
[JDK_INSTALLED_PATH]\jre\lib;
[JDK_INSTALLED_PATH]\lib;
```

- JDBC ライブラリフォルダに、SQLJDBC4.jar ファイルのみがあることを確認します。このフォルダから、他の sqljdbc*.jar ファイルを削除します (または他のフォルダにコピーします)。ドライバをプログラムの一部として追加する場合は、使用するドライバとして SQLJDBC4.jar のみを追加します。
- sqljdbc_auth.dll ファイルをアプリケーションのフォルダにコピーします。

> **Note**
>
> 32 ビットの Java Virtual Machine (JVM) を実行している場合は、オペレーティングシステムが x64 バージョンであっても、x86 フォルダの sqljdbc_auth.dll ファイルを使用します。64 ビットの JVM を x64 プロセッサで実行している場合は、x64 フォルダの sqljdbc_auth.dll ファイルを使用します。

SQL Server データベースに接続するときは、[Windows Authentication] (Windows 認証) または [SQL Server Authentication] (SQL Server 認証) のどちらかを [Authentication] (認証) オプションで選択できます。

ソースとしての SQL Server への接続

以下の手順を使用して、Microsoft SQL Server ソースデータベースを AWS Schema Conversion Tool に接続します。

Microsoft SQL Server ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Microsoft SQL Server]、次に [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。

2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。

- Microsoft SQL Server ソースデータベースの接続情報を手動で入力するには、次の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームサービス (DNS) 名または IP アドレスを入力します。 ソースの SQL Server データベースには、IPv6 アドレスプロトコルを使用して接続できます。そのためには、次の例に示すように IP アドレスの入力に角括弧を使用することを確認します。 `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
Instance name	SQL Server データベースのインスタンス名を入力します。インスタンス名を見つけるには、SQL Server データベースに対してクエリ `SELECT @@servername;` を実行します。
認証	[Windows Authentication] (Windows 認証) および [SQL Server Authentication] (SQL Server 認証) から認証タイプを選択します。

パラメータ	アクション
[User name] (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソース データベース サーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー証明書を信頼する]: サーバー証明書を信頼するには、このオプションを選択します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。この場所を [グローバル設定] セクションに表示するには、必ず追加してください。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションを有効にすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
Sql Server Driver Path	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

パラメータ	アクション
Windows 認証ライブラリ	sqljdbc_auth.dll ファイルへのパスを入力します。デフォルトでは、このファイルは次の場所にインストールされます。 *<installation directory of the JDBC driver>*sqljdbc_*<version>* *<language>* \auth\

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

SQL Server から MySQL への変換

変換した MySQL コードで Microsoft SQL Server データベース関数をエミュレートするには、AWS SCT で SQL Server から MySQL への拡張パックを使用します。拡張パックの詳細については、「AWS SCT 拡張パックの使用」を参照してください。

トピック

- MySQL をターゲットデータベースとする場合の権限
- SQL Server から MySQL への変換設定
- 移行に関する考慮事項

MySQL をターゲットデータベースとする場合の権限

ターゲットとして MySQL に必要な権限を以下に示します。

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*

- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- INSERT, UPDATE ON AWS_SQLSERVER_EXT.*
- INSERT, UPDATE, DELETE ON AWS_SQLSERVER_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_SQLSERVER_EXT_DATA.*

次のコード例を使用すると、データベースユーザーを作成して、権限を付与できます。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_SQLSERVER_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_SQLSERVER_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_SQLSERVER_EXT_DATA.* TO 'user_name';
```

上記の例の *user_name* は使用するユーザー名に置き換えます。次に、*your_password* を安全なパスワードに置き換えます。

バージョン 5.7 以前の MySQL データベースをターゲットとして使用する場合は、次のコマンドを実行します。MySQL データベースバージョン 8.0 以降では、このコマンドは廃止されました。

```
GRANT SELECT ON mysql.proc TO 'user_name';
```

Amazon RDS for MySQL または Aurora MySQL をターゲットとして使用するには、`lower_case_table_names` パラメータを 1 と設定します。この値は、MySQL サーバーが

テーブル、インデックス、トリガー、データベースなどのオブジェクト名の識別子を、大文字と小文字を区別せずに処理することを意味します。ターゲットインスタンスでバイナリログを有効にしている場合は、log_bin_trust_function_creators パラメータを 1 と設定します。この場合、ストアドファンクションを作成するのに、DETERMINISTIC 特性、READS SQL DATA 特性、NO SQL 特性を使用する必要はありません。これらのパラメータを設定するには、新しい DB パラメータグループを作成するか、既存の DB パラメータグループを変更します。

SQL Server から MySQL への変換設定

AWS SCT で SQL Server から MySQL への変換設定を編集するには、[設定] を選択し、[変換設定] を選択します。上のリストから [SQL Server] を選択し、[SQL Server — MySQL] を選択します。AWS SCT に、SQL Server から MySQL への変換に使用可能なすべての設定が表示されます。

AWS SCT での SQL Server から MySQL への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- ソース SQL Server データベースが EXEC の出力をテーブルに保存できるようにします。AWS SCT は、一時テーブルと、この機能をエミュレートする追加のプロシージャを作成します。このエミュレーションを使用するには、[オープンデータセットを処理するための追加ルーチンを作成する] を選択します。

移行に関する考慮事項

SQL Server スキーマを MySQL に移行する際は以下の点を考慮してください。

- MySQL は MERGE ステートメントをサポートしていません。ただし、AWS SCT は、INSERT ON DUPLICATE KEY 句と UPDATE FROM and DELETE FROM ステートメントを使用して、変換中に MERGE ステートメントをエミュレートできます。

  INSERT ON DUPLICATE KEY を使用して正しくエミュレーションするために、一意の制約またはプライマリキーがターゲット MySQL データベースで存在することを確認してください。

- GOTO ステートメントとラベルを使用して、ステートメントを実行する順序を変更できます。GOTO ステートメントに続くすべての Transact-SQL ステートメントはスキップされ、ラベルで処理が継続されます。GOTO ステートメントとラベルは、プロシージャ、バッチ、またはステートメント ブロック内のどこでも使用できます。GOTO ステートメントをネストすることもできます。

  MySQL は GOTO ステートメントを使用しません。AWS SCT が GOTO ステートメントが含まれるコードを変換する場合、ステートメントは BEGIN…END または LOOP…END LOOP ステートメントを使用するように変換されます。次の表は、AWS SCT が GOTO ステートメントを変換する方法の一例です。

SQL Server ステートメント	MySQL ステートメント

BEGIN
....
statement1;
....
GOTO label1;
statement2;
....
label1:
Statement3;
....
END
	BEGIN
	label1:
	BEGIN
	....
	statement1;
	....
	LEAVE label1;
	statement2;
	....
	END;
	Statement3;
	....
	END

SQL Server ステートメント	MySQL ステートメント
BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	BEGIN .... statement1; .... label1: LOOP statement2; .... ITERATE label1; LEAVE label1; END LOOP; statement3; .... statement4; .... END

BEGIN
....
statement1;
....
label1:
statement2;
....
statement3;
....
statement4;
....
END
	BEGIN
	....
	statement1;
	....
	label1:
	BEGIN
	statement2;
	....
	statement3;
	....
	statement4;
	....
	END;
	END

- MySQL は複数ステートメントのテーブル値関数をサポートしていません。AWS SCT では、一時テーブルを作成し、これらの一時テーブルを使用するようにステートメントを書き換えることで、変換中にテーブル値関数をシミュレートします。

SQL Server から PostgreSQL への変換

AWS SCT で SQL サーバーから PostgreSQL への拡張パックを使用できます。この拡張パックは、変換された PostgreSQL コード内の SQL Server データベース関数をエミュレートします。SQL Server to PostgreSQL 拡張パックを使用すると、SQL Server エージェントと SQL Server データベースメールをエミュレートできます。拡張パックの詳細については、「AWS SCT 拡張パックの使用」を参照してください。

トピック

ターゲットデータベースとしての PostgreSQL の権限

PostgreSQL をターゲットとして使用するには、AWS SCT に CREATE ON DATABASE 権限が必要です。ターゲット PostgreSQL データベースごとにこの権限を必ず付与してください。

変換されたパブリックシノニムを使用するには、データベースのデフォルト検索パスを "$user", public_synonyms, public に変更します。

次のコード例を使用すると、データベースユーザーを作成して、権限を付与できます。

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

上記の例の *user_name* は、使用するユーザー名に置き換えます。[*db_name*] をターゲットデータベースの名前に置き換えます。最後に、[*your_password*] を安全なパスワードに置き換えます。

PostgreSQL では、スキーマを削除できるのはスキーマ所有者または superuser のみです。スキーマ所有者が一部のオブジェクトを所有していない場合でも、スキーマとスキーマに含まれるすべてのオブジェクトを削除できます。

異なるユーザーを使用して異なるスキーマを変換してターゲットデータベースに適用すると、AWS SCT がスキーマを削除できないときにエラーメッセージが表示されることがあります。このエラーメッセージを回避するには、`superuser` ロールを使用します。

SQL Server から PostgreSQL への変換設定

SQL Server から PostgreSQL への変換設定を編集するには、[設定] を選択し、次に [変換設定] を選択します。上のリストから [SQL Server] を選択し、[SQL Server — PostgreSQL] を選択します。AWS SCT に、SQL Server から PostgreSQL への変換に使用可能なすべての設定が表示されます。

AWS SCT の SQL Server から PostgreSQL への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- SQL Server の異なるテーブルで同じ名前のインデックスを使用できるようにする。

  PostgreSQL では、スキーマで使用するインデックス名はすべて一意でなければなりません。AWS SCT ですべてのインデックスに固有の名前が生成されるようにするには、[インデックスの固有の名前を生成] を選択します。
- SQL Server プロシージャを PostgreSQL 関数に変換する。

  PostgreSQL バージョン 10 以前のバージョンはプロシージャをサポートしていません。PostgreSQL でのプロシージャの使用に慣れていない場合は、AWS SCT がプロシージャを関数に変換できます。このためには、[プロシージャを関数に変換] を選択します。
- `EXEC` の出力をテーブルでエミュレートする。

  ソース SQL Server データベースが `EXEC` の出力をテーブルに保存できるようにします。AWS SCT は、一時テーブルと、この機能をエミュレートする追加のプロシージャを作成します。このエミュレーションを使用するには、[オープンデータセットを処理するための追加ルーチンを作成する] を選択します。

- 変換されるコード内のスキーマ名に使用するテンプレートを定義する。[スキーマ名生成テンプレート] では、次のオプションのいずれかを選択します。
  - [<source_db>] – SQL Server のデータベース名を PostgreSQL のスキーマ名として使用する。
  - [<source_schema>] PostgreSQL のスキーマ名として SQL サーバのスキーマ名を使用します。
  - [<source_db>_<schema>] – SQL Server データベースとスキーマ名の組み合わせを PostgreSQL のスキーマ名として使用します。
- ソースオブジェクト名の大文字と小文字を区別しない。

  オブジェクト名が小文字に変換されないようにするには、[大文字と小文字を区別する操作では小文字へのキャストを避ける] を選択します。このオプションは、ターゲットデータベースで大文字と小文字の区別オプションをオンにした場合にのみ適用されます。
- ソースデータベースのパラメータ名をそのまま使用する。

  変換されたコード内のパラメータ名に二重引用符を追加するには、[元のパラメータ名を保持] を選択します。

SQL Server パーティションから PostgreSQL バージョン 10 パーティションへの変換

Microsoft SQL Server データベースを Amazon Aurora PostgreSQL 互換エディション (Aurora PostgreSQL) または Amazon Relational Database Service for PostgreSQL (Amazon RDS for PostgreSQL) に変換するときは、次の点に注意してください。

SQL Server で、パーティション関数を持つパーティションを作成します。SQL Server の一部のテーブルから PostgreSQL バージョン 10 のパーティション分割テーブルに変換する際は、いくつか潜在的な問題があることに注意してください。

- SQL Server では、NOT NULL 制約を使用しない列を使用してテーブルをパーティション分割できます。この場合、すべての NULL 値は左端のパーティションに移動されます。PostgreSQL は RANGE パーティションの NULL 値をサポートしていません。
- SQL Server では、パーティション分割されたテーブルでプライマリキーや固有キーを作成できます。PostgreSQL の場合、プライマリーキーや固有キーをパーティションごと直接作成します。したがって、PostgreSQL への移行時に親テーブルから PRIMARY または UNIQUE KEY 制約を削除する必要があります。生成されたキー名は `<original_key_name>_<partition_number>` の形式です。
- SQL Server では、パーティション分割されたテーブルから、またはテーブルに、外部キー制約を作成できます。PostgreSQL はパーティション分割されたテーブルを参照する外部キーをサポート

していません。また、PostgreSQL は分割されたテーブルから別のテーブルへの外部キーの参照をサポートしていません。

- SQL Server では、パーティション分割されたテーブルでインデックスを作成できます。PostgreSQL の場合、パーティションごとにインデックスを直接作成する必要があります。したがって、インデックスは PostgreSQL への移行時に親テーブルから削除する必要があります。生成されたインデックス名は `<original_index_name>_<partition_number>` の形式です。
- PostgreSQL ではパーティション分割されたインデックスがサポートされていません。

移行に関する考慮事項

SQL Server スキーマを PostgreSQL に移行する際は以下の点を考慮してください。

- PostgreSQL では、スキーマ内のすべてのオブジェクト名は、インデックスを含めて一意である必要があります。インデックス名は、ベーステーブルのスキーマで一意である必要があります。SQL Server では、異なるテーブルでは同じインデックス名を使用できます。

  インデックス名の一意性を確保するために、AWS SCT は、インデックス名が一意ではない場合に一意のインデックスを生成するオプションを提供します。これを行うには、プロジェクトのプロパティで一意のインデックス名を生成するオプションを選択します。デフォルトでは、この機能は有効になっています。このオプションが有効になっている場合、一意のインデックス名が IX_table_name_index_name 形式を使用して作成されます。このオプションが無効になっている場合、インデックス名は変更されません。
- GOTO ステートメントとラベルを使用して、ステートメントを実行する順序を変更できます。GOTO ステートメントに続くすべての Transact-SQL ステートメントはスキップされ、ラベルで処理が継続されます。GOTO ステートメントとラベルはプロシージャ、バッチ、またはステートメントブロックの任意の場所で使用できます。GOTO ステートメントをネストすることもできます。

  PostgreSQL は GOTO ステートメントを使用しません。AWS SCT が GOTO ステートメントが含まれるコードを変換する場合、ステートメントは BEGIN...END または LOOP...END LOOP ステートメントを使用するように変換されます。次の表は、AWS SCT が GOTO ステートメントを変換する方法の一例です。

SQL Server の GOTO ステートメントと、変換された PostgreSQL ステートメント

SQL Server ステートメント	PostgreSQL ステートメント

BEGIN
....
statement1;
....
GOTO label1;
statement2;
....
label1:
Statement3;
....
END
	BEGIN
	label1:
	BEGIN
	....
	statement1;
	....
	EXIT label1;
	statement2;
	....
	END;
	Statement3;
	....
	END

BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	BEGIN .... statement1; .... label1: LOOP statement2; .... CONTINUE label1; EXIT label1; END LOOP; statement3; .... statement4; .... END

SQL Server ステートメント	PostgreSQL ステートメント

```
BEGIN
   ....
   statement1;
   ....
   label1:
   statement2;
   ....
   statement3;
   ....
   statement4;
   ....
END
```

```
BEGIN
   ....
   statement1;
   ....
   label1:
   BEGIN
    statement2;
    ....
    statement3;
    ....
    statement4;
    ....
   END;
END
```

- PostgreSQL は MERGE ステートメントをサポートしていません。AWS SCT は MERGE ステートメントの動作を次の方法でエミュレートします。
  - INSERT の ON CONFLICT 句。
  - UPDATE FROM DML ステートメント (例: WHEN NOT MATCHED 句を指定しない MERGE)。
  - CURSOR (例: DELETE 句を指定した MERGE)、または複雑な MERGE ON 条件ステートメント。
- AWS SCT は Amazon RDS がターゲットの場合にデータベーストリガーをオブジェクトツリーに追加できます。
- AWS SCT は Amazon RDS がターゲットの場合にサーバーレベルトリガーをオブジェクトツリーに追加できます。
- SQL Server は `deleted` および `inserted` テーブルを自動的に作成および管理します。メモリに常駐するこれらの一時テーブルを使用して、特定のデータ変更の影響をテストしたり、DML トリガーアクションの条件を設定したりできます。AWS SCT は、DML トリガーステートメント内でのこれらのテーブルの使用方法を変換できます。
- AWS SCT は Amazon RDS がターゲットの場合にリンクサーバーをオブジェクトツリーに追加できます。
- Microsoft SQL Server から PostgreSQL に移行する場合、組み込み SUSER_SNAME 関数は次のように変換されます。
  - SUSER_SNAME - セキュリティ識別番号 (SID) に関連付けられたログイン名を返します。

  - SUSER_SNAME(<server_user_sid>) – サポート外です。
  - SUSER_SNAME() CURRENT_USER – 現在の実行コンテキストのユーザー名を返します。
  - SUSER_SNAME(NULL) – NULL が返されます。
- テーブル値関数の変換がサポートされています。テーブル値関数はテーブルを返し、クエリ内のテーブルに代わることができます。
- PATINDEX は、すべての有効なテキストおよび文字データ型で指定された式でパターンが最初に出現する開始位置を返します。パターンが見つからない場合は、ゼロを返します。SQL Server から Amazon RDS for PostgreSQL に変換する場合、AWS SCT は PATINDEX を使用するアプリケーションコードを、aws_sqlserver_ext.patindex (<パターン文字>、<可変の式の文字>) に置き換えます。
- SQL Server では、ユーザー定義のテーブルタイプは、テーブル構造の定義を表すタイプです。ユーザー定義のテーブル型を使用して、ストアドプロシージャまたは関数のテーブル値パラメータを宣言します。また、ユーザー定義テーブル型を使用して、バッチまたはストアドプロシージャまたは関数の本体で使用するテーブル変数を宣言することもできます。AWS SCT は一時テーブルを作成することで PostgreSQL 内のこの型を エミュレートします。

AWS SCT は、SQL Server から PostgreSQL に変換するときに、SQL Server システムオブジェクトを PostgreSQL で認識可能なオブジェクトに変換します。次の表に、システムオブジェクトの変換方法を示します。

MS SQL Server ユースケース	PostgreSQL の置換
SYS.SCHEMAS	AWS_SQLSERVER_EXT.SYS_SCHEMAS
SYS.TABLES	AWS_SQLSERVER_EXT.SYS_TABLES
SYS.VIEWS	AWS_SQLSERVER_EXT.SYS_VIEWS
SYS.ALL_VIEWS	AWS_SQLSERVER_EXT.SYS_ALL_VIEWS
SYS.TYPES	AWS_SQLSERVER_EXT.SYS_TYPES
SYS.COLUMNS	AWS_SQLSERVER_EXT.SYS_COLUMNS
SYS.ALL_COLUMNS	AWS_SQLSERVER_EXT.SYS_ALL_COLUMNS

MS SQL Server ユースケース	PostgreSQL の置換
SYS.FOREIGN_KEYS	AWS_SQLSERVER_EXT.SYS_FOREIGN_KEYS
SYS.SYSFOREIGNKEYS	AWS_SQLSERVER_EXT.SYS_SYSFOREIGNKEYS
SYS.FOREIGN_KEY_COLUMNS	AWS_SQLSERVER_EXT.SYS_FOREIGN_KEY_COLUMNS
SYS.KEY_CONSTRAINTS	AWS_SQLSERVER_EXT.SYS_KEY_CONSTRAINTS
SYS.IDENTITY_COLUMNS	AWS_SQLSERVER_EXT.SYS_IDENTITY_COLUMNS
SYS.PROCEDURES	AWS_SQLSERVER_EXT.SYS_PROCEDURES
SYS.INDEXES	AWS_SQLSERVER_EXT.SYS_INDEXES
SYS.SYSINDEXES	AWS_SQLSERVER_EXT.SYS_SYSINDEXES
SYS.OBJECTS	AWS_SQLSERVER_EXT.SYS_OBJECTS
SYS.ALL_OBJECTS	AWS_SQLSERVER_EXT.SYS_ALL_OBJECTS
SYS.SYSOBJECTS	AWS_SQLSERVER_EXT.SYS_SYSOBJECTS
SYS.SQL_MODULES	AWS_SQLSERVER_EXT.SYS_SQL_MODULES
SYS.DATABASES	AWS_SQLSERVER_EXT.SYS_DATABASES
INFORMATION_SCHEMA.SCHEMATA	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_SCHEMATA
INFORMATION_SCHEMA.VIEWS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_VIEWS
INFORMATION_SCHEMA.TABLES	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLES

MS SQL Server ユースケース	PostgreSQL の置換
INFORMATION_SCHEMA.COLUMNS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_COLUMNS
INFORMATION_SCHEMA.CHECK_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CHECK_CONSTRAINTS
INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_REFERENTIAL_CONSTRAINTS
INFORMATION_SCHEMA.TABLE_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLE_CONSTRAINTS
INFORMATION_SCHEMA.KEY_COLUMN_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_KEY_COLUMN_USAGE
INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_TABLE_USAGE
INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_COLUMN_USAGE
INFORMATION_SCHEMA.ROUTINES	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_ROUTINES
SYS.SYSPROCESSES	AWS_SQLSERVER_EXT.SYS_SYSPROCESSES
sys.system_objects	AWS_SQLSERVER_EXT.SYS_SYSTEM_OBJECTS

AWS SCT 拡張パックを使用して PostgreSQL で SQL Server エージェントをエミュレートする

SQL Server エージェントは、SQL Server ジョブを実行する Microsoft Windows サービスです。SQL Server エージェントは、スケジュールに従って、特定のイベントに応答して、またはオンデマンド

でジョブを実行します。SQL Server エージェントの詳細については、『Microsoft 技術ドキュメント』を参照してください。

PostgreSQL エージェントに相当するものはありません。SQL Server エージェントの機能をエミュレートするために、AWS SCT は拡張パックを作成します。この拡張パックは AWS Lambda と Amazon CloudWatch を使用しています。AWS Lambda は、スケジュールの管理やジョブの実行に使用するインターフェースを実装します。Amazon CloudWatch はスケジュールルールを管理します。

AWS Lambda と Amazon CloudWatch は JSON パラメータを使用してやり取りします。この JSON パラメータは次の構造があります。

```
{
    "mode": mode,
    "parameters": {
        list of parameters
    },
    "callback": procedure name
}
```

前の例では、*mode* はタスクのタイプであり、*list of parameters* タスクのタイプによって決まるパラメータのセットです。また、*procedure name* は、タスクが完了した後に実行されるプロシージャの名前でもあります。

AWS SCT は 1 つの Lambda 関数を使用してジョブの制御と実行を行います。CloudWatch ルールはジョブの実行を開始し、ジョブを開始するために必要な情報を提供します。CloudWatch ルールがトリガーされると、ルールのパラメータを使用して Lambda 関数が開始されます。

プロシージャを呼び出す簡単なジョブを作成するには、次の形式を使用します。

```
{
    "mode": "run_job",
    "parameters": {
        "vendor": "mysql",
        "cmd": "lambda_db.nightly_job"
    }
}
```

複数のステップを含むジョブを作成するには、次の形式を使用します。

```
{
```

```
    "mode": "run_job",
    "parameters": {
        "job_name": "Job1",
        "enabled": "true",
        "start_step_id": 1,
        "notify_level_email": [0|1|2|3],
        "notify_email": email,
        "delete_level": [0|1|2|3],
        "job_callback": "ProcCallBackJob(job_name, code, message)",
        "step_callback": "ProcCallBackStep(job_name, step_id, code, message)"
    },
    "steps": [
        {
            "id":1,
            "cmd": "ProcStep1",
            "cmdexec_success_code": 0,
            "on_success_action": [|2|3|4],
            "on_success_step_id": 1,
            "on_fail_action": 0,
            "on_fail_step_id": 0,
            "retry_attempts": number,
            "retry_interval": number
        },
        {
            "id":2,
            "cmd": "ProcStep2",
            "cmdexec_success_code": 0,
            "on_success_action": [1|2|3|4],
            "on_success_step_id": 0,
            "on_fail_action": 0,
            "on_fail_step_id": 0,
            "retry_attempts": number,
            "retry_interval": number
        },
        ...
]
}
```

PostgreSQL での SQL Server エージェントの動作をエミュレートするために、AWS SCT 拡張パックでは以下のテーブルとプロシージャも作成します。

PostgreSQL で SQL サーバーエージェントをエミュレートするテーブル

SQL Server エージェントをエミュレートするために、拡張パックは次のテーブルを使用します。

sysjobs

ジョブに関する情報を格納します。

sysjobsteps

ジョブのステップに関する情報を格納します。

sysschedules

ジョブスケジュールに関する情報を格納します。

sysjobschedules

個々のジョブのスケジュール情報を格納します。

sysjobhistory

スケジュールされたジョブの実行に関する情報を格納します。

PostgreSQL で SQL サーバーエージェントをエミュレートするプロシージャ

SQL Server エージェントをエミュレートするために、拡張パックでは以下の手順を使用します。

sp_add_job

新しいジョブを追加します。

sp_add_jobstep

ジョブにステップを追加します。

sp_add_schedule

Amazon CloudWatch に新しいスケジュールルールを作成します。このスケジュールは任意の数のジョブで使用できます。

sp_attach_schedule

選択したジョブのスケジュールを設定します。

sp_add_jobschedule

Amazon CloudWatch でジョブのスケジュールルールを作成し、このルールのターゲットを設定します。

sp_update_job

以前に作成したジョブの属性を更新します。

sp_update_jobstep

ジョブ内のステップの属性を更新します。

sp_update_schedule

Amazon CloudWatch のスケジュールルールの属性を更新します。

sp_update_jobschedule

指定ジョブのスケジュールの属性を更新します。

sp_delete_job

ジョブを削除します。

sp_delete_jobstep

ジョブからジョブステップを削除します。

sp_delete_schedule

スケジュールを削除します。

sp_delete_jobschedule

指定されたジョブのスケジュールルールを Amazon CloudWatch から削除します。

sp_detach_schedule

スケジュールとジョブの関連付けを削除します。

get_jobs、update_job

AWS Elastic Beanstalk と相互作用する内部プロシージャ。

sp_verify_job_date、sp_verify_job_time、sp_verify_job、sp_verify_jobstep、sp_verify_schedule、sp_verify_j

設定を確認する内部プロシージャ。

PostgreSQL で SQL サーバーエージェントをエミュレートするプロシージャの構文

拡張パック内の `aws_sqlserver_ext.sp_add_job` プロシージャは、`msdb.dbo.sp_add_job` プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_name varchar,
par_enabled smallint = 1,
```

```
par_description varchar = NULL::character varying,
par_start_step_id integer = 1,
par_category_name varchar = NULL::character varying,
par_category_id integer = NULL::integer,
par_owner_login_name varchar = NULL::character varying,
par_notify_level_eventlog integer = 2,
par_notify_level_email integer = 0,
par_notify_level_netsend integer = 0,
par_notify_level_page integer = 0,
par_notify_email_operator_name varchar = NULL::character varying,
par_notify_netsend_operator_name varchar = NULL::character varying,
par_notify_page_operator_name varchar = NULL::character varying,
par_delete_level integer = 0,
inout par_job_id integer = NULL::integer,
par_originating_server varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_add_jobstep プロシージャは、msdb.dbo.sp_add_jobstep プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_step_id integer = NULL::integer,
par_step_name varchar = NULL::character varying,
par_subsystem varchar = 'TSQL'::bpchar,
par_command text = NULL::text,
par_additional_parameters text = NULL::text,
par_cmdexec_success_code integer = 0,
par_on_success_action smallint = 1,
par_on_success_step_id integer = 0,
par_on_fail_action smallint = 2,
par_on_fail_step_id integer = 0,
par_server varchar = NULL::character varying,
par_database_name varchar = NULL::character varying,
par_database_user_name varchar = NULL::character varying,
par_retry_attempts integer = 0,
par_retry_interval integer = 0,
par_os_run_priority integer = 0,
par_output_file_name varchar = NULL::character varying,
par_flags integer = 0,
par_proxy_id integer = NULL::integer,
par_proxy_name varchar = NULL::character varying,
```

```
inout par_step_uid char = NULL::bpchar,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_add_schedule プロシージャは、msdb.dbo.sp_add_schedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_schedule_name varchar,
par_enabled smallint = 1,
par_freq_type integer = 0,
par_freq_interval integer = 0,
par_freq_subday_type integer = 0,
par_freq_subday_interval integer = 0,
par_freq_relative_interval integer = 0,
par_freq_recurrence_factor integer = 0,
par_active_start_date integer = NULL::integer,
par_active_end_date integer = 99991231,
par_active_start_time integer = 0,
par_active_end_time integer = 235959,
par_owner_login_name varchar = NULL::character varying,
*inout par_schedule_uid char = NULL::bpchar,*
inout par_schedule_id integer = NULL::integer,
par_originating_server varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_attach_schedule プロシージャは、msdb.dbo.sp_attach_schedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_automatic_post smallint = 1,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_add_jobschedule プロシージャは、msdb.dbo.sp_add_jobschedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer,
```

```
par_job_name varchar = NULL::character varying,
par_name varchar = NULL::character varying,
par_enabled smallint = 1,
par_freq_type integer = 1,
par_freq_interval integer = 0,
par_freq_subday_type integer = 0,
par_freq_subday_interval integer = 0,
par_freq_relative_interval integer = 0,
par_freq_recurrence_factor integer = 0,
par_active_start_date integer = NULL::integer,
par_active_end_date integer = 99991231,
par_active_start_time integer = 0,
par_active_end_time integer = 235959,
inout par_schedule_id integer = NULL::integer,
par_automatic_post smallint = 1,
inout par_schedule_uid char = NULL::bpchar,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_delete_job プロシージャは、msdb.dbo.sp_delete_job プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_originating_server varchar = NULL::character varying,
par_delete_history smallint = 1,
par_delete_unused_schedule smallint = 1,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_delete_jobstep プロシージャは、msdb.dbo.sp_delete_jobstep プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_step_id integer = NULL::integer,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_delete_jobschedule プロシージャは、msdb.dbo.sp_delete_jobschedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_name varchar = NULL::character varying,
par_keep_schedule integer = 0,
par_automatic_post smallint = 1,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_delete_schedule プロシージャは、msdb.dbo.sp_delete_schedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_force_delete smallint = 0,
par_automatic_post smallint = 1,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_detach_schedule プロシージャは、msdb.dbo.sp_detach_schedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_delete_unused_schedule smallint = 0,
par_automatic_post smallint = 1,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_update_job プロシージャは、msdb.dbo.sp_update_job プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_description varchar = NULL::character varying
par_start_step_id integer = NULL::integer
par_category_name varchar = NULL::character varying
```

```
par_owner_login_name varchar = NULL::character varying
par_notify_level_eventlog integer = NULL::integer
par_notify_level_email integer = NULL::integer
par_notify_level_netsend integer = NULL::integer
par_notify_level_page integer = NULL::integer
par_notify_email_operator_name varchar = NULL::character varying
par_notify_netsend_operator_name varchar = NULL::character varying
par_notify_page_operator_name varchar = NULL::character varying
par_delete_level integer = NULL::integer
par_automatic_post smallint = 1
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_update_jobschedule プロシージャは、msdb.dbo.sp_update_jobschedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_freq_type integer = NULL::integer
par_freq_interval integer = NULL::integer
par_freq_subday_type integer = NULL::integer
par_freq_subday_interval integer = NULL::integer
par_freq_relative_interval integer = NULL::integer
par_freq_recurrence_factor integer = NULL::integer
par_active_start_date integer = NULL::integer
par_active_end_date integer = NULL::integer
par_active_start_time integer = NULL::integer
                par_active_end_time integer = NULL::integer
par_automatic_post smallint = 1
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_update_jobstep プロシージャは、msdb.dbo.sp_update_jobstep プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_step_id integer = NULL::integer
par_step_name varchar = NULL::character varying
```

```
par_subsystem varchar = NULL::character varying
par_command text = NULL::text
par_additional_parameters text = NULL::text
par_cmdexec_success_code integer = NULL::integer
par_on_success_action smallint = NULL::smallint
par_on_success_step_id integer = NULL::integer
par_on_fail_action smallint = NULL::smallint
par_on_fail_step_id integer = NULL::integer
par_server varchar = NULL::character varying
par_database_name varchar = NULL::character varying
par_database_user_name varchar = NULL::character varying
par_retry_attempts integer = NULL::integer
par_retry_interval integer = NULL::integer
par_os_run_priority integer = NULL::integer
par_output_file_name varchar = NULL::character varying
par_flags integer = NULL::integer
par_proxy_id integer = NULL::integer
par_proxy_name varchar = NULL::character varying
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sp_update_schedule プロシージャは、msdb.dbo.sp_update_schedule プロシージャをエミュレートします。SQL Server エージェントプロシージャの詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_schedule_id integer = NULL::integer
par_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_freq_type integer = NULL::integer
par_freq_interval integer = NULL::integer
par_freq_subday_type integer = NULL::integer
par_freq_subday_interval integer = NULL::integer
par_freq_relative_interval integer = NULL::integer
par_freq_recurrence_factor integer = NULL::integer
par_active_start_date integer = NULL::integer
par_active_end_date integer = NULL::integer
par_active_start_time integer = NULL::integer
par_active_end_time integer = NULL::integer
par_owner_login_name varchar = NULL::character varying
par_automatic_post smallint = 1
out returncode integer
```

PostgreSQL で SQL Server エージェントをエミュレートするプロシージャを使用する場合の例

新しいジョブを追加するには、次に示す aws_sqlserver_ext.sp_add_job プロシージャに従います。

```
SELECT * FROM aws_sqlserver_ext.sp_add_job (
    par_job_name := 'test_job',
    par_enabled := 1::smallint,
    par_start_step_id := 1::integer,
    par_category_name := '[Uncategorized (Local)]',
    par_owner_login_name := 'sa');
```

新しいジョブステップを追加するには、次に示す aws_sqlserver_ext.sp_add_jobstep プロシージャに従います。

```
SELECT * FROM aws_sqlserver_ext.sp_add_jobstep (
    par_job_name := 'test_job',
    par_step_id := 1::smallint,
    par_step_name := 'test_job_step1',
    par_subsystem := 'TSQL',
    par_command := 'EXECUTE [dbo].[PROC_TEST_JOB_STEP1];',
    par_server := NULL,
    par_database_name := 'GOLD_TEST_SS');
```

簡単なスケジュールを追加するには、次に示す aws_sqlserver_ext.sp_add_schedule プロシージャに従います。

```
SELECT * FROM aws_sqlserver_ext.sp_add_schedule(
    par_schedule_name := 'RunOnce',
    par_freq_type := 1,
    par_active_start_time := 233000);
```

ジョブのスケジュールを設定するには、次に示す aws_sqlserver_ext.sp_attach_schedule プロシージャに従います。

```
SELECT * FROM aws_sqlserver_ext.sp_attach_schedule (
    par_job_name := 'test_job',
    par_schedule_name := 'NightlyJobs');
```

ジョブのスケジュールを作成するには、次に示す aws_sqlserver_ext.sp_add_jobschedule プロシージャに従います。

```
SELECT * FROM aws_sqlserver_ext.sp_add_jobschedule (
    par_job_name := 'test_job2',
    par_name := 'test_schedule2',
    par_enabled := 1::smallint,
    par_freq_type := 4,
    par_freq_interval := 1,
    par_freq_subday_type := 4,
    par_freq_subday_interval := 1,
    par_freq_relative_interval := 0,
    par_freq_recurrence_factor := 0,
    par_active_start_date := 20100801,
    par_active_end_date := 99991231,
    par_active_start_time := 0,
    par_active_end_time := 0);
```

PostgreSQL で SQL サーバーエージェントをエミュレートするためのユースケースの例

ソースデータベースコードで SQL Server エージェントを使用してジョブを実行している場合、AWS SCT はSQL Server から PostgreSQL への拡張パックを使用して、このコードを PostgreSQL に変換できます。拡張パックは AWS Lambda 関数を使用して SQL Server エージェントの動作をエミュレートします。

新しい AWS Lambda 関数を作成するか、既存の関数を登録できます。

新しい AWS Lambda 関数を作成するには

1. AWS SCT のターゲットデータベースのツリーで、コンテキスト (右クリック) メニューを開き、[次に拡張パックの適用] () を選択して、[PostgreSQL] を選択します。

   拡張パックウィザードが表示されます。
2. [SQL Server エージェントエミュレーションサービス] タブで、次の操作を行います。

   - [AWS Lambda 関数の作成] を選択します。
   - [データベースログイン] には、ターゲットデータベースユーザーの名前を入力します。
   - [データベースパスワード] には、前のステップで入力したユーザー名のパスワードを入力します。
   - [Python ライブラリフォルダ] には、Python ライブラリフォルダへのパスを入力します。
   - [AWS Lambda 関数の作成] を選択し、[次へ] を選択します。

以前にデプロイした AWS Lambda 関数を登録するには

- ターゲットデータベースで次のスクリプトを実行します。

```
SELECT
    FROM aws_sqlserver_ext.set_service_setting(
        p_service := 'JOB',
        p_setting := 'LAMBDA_ARN',
        p_value := ARN)
```

前述の例では、*ARN* はデプロイされた AWS Lambda 関数の Amazon リソースネーム (ARN) です。

次の例では、1 つのステップで構成されるシンプルなタスクを作成します。このタスクでは、5 分ごとに以前に作成された job_example 関数が実行されます。この job_example_table 関数はレコードをテーブルに挿入します。

この簡単なタスクを作成するには

1. 以下の aws_sqlserver_ext.sp_add_job 関数を使用してジョブを作成します。

```
SELECT
    FROM aws_sqlserver_ext.sp_add_job (
        par_job_name := 'test_simple_job');
```

2. 次に示す aws_sqlserver_ext.sp_add_jobstep 関数を使用してジョブステップを作成します。

```
SELECT
    FROM aws_sqlserver_ext.sp_add_jobstep (
        par_job_name := 'test_simple_job',
        par_step_name := 'test_simple_job_step1',
        par_command := 'PERFORM job_simple_example;');
```

ジョブステップは関数の実行内容を指定します。

3. 次に示すように、aws_sqlserver_ext.sp_add_jobschedule 関数を使用してジョブのスケジューラーを作成します。

```
SELECT
```

```
    FROM aws_sqlserver_ext.sp_add_jobschedule (
        par_job_name := 'test_simple_job',
        par_name := 'test_schedule',
        par_freq_type := 4, /* Daily */
        par_freq_interval := 1, /* frequency_interval is unused */
        par_freq_subday_type := 4, /* Minutes */
        par_freq_subday_interval := 5 /* 5 minutes */);
```

ジョブステップは関数の実行内容を指定します。

このジョブを削除するには、以下の aws_sqlserver_ext.sp_delete_job 関数を使用します。

```
PERFORM aws_sqlserver_ext.sp_delete_job(
    par_job_name := 'PeriodicJob1'::character varying,
    par_delete_history := 1::smallint,
    par_delete_unused_schedule := 1::smallint);
```

AWS SCT 拡張パックを使用して PostgreSQL で SQL Server データベースメールをエミュレートする

SQL Server データベース メールを使用すると、SQL Server データベース エンジンまたは Azure SQL Managed Instance からユーザーに電子メール メッセージを送信できます。これらのメール メッセージには、クエリ結果を含めたり、ネットワーク上の任意のリソースからのファイルを含めることができます。SQL Server データベースメールの詳細については、『Microsoft 技術ドキュメント』を参照してください。

PostgreSQL には SQL Server データベースメールに相当するものはありません。SQL Server データベースメール機能をエミュレートするために、AWS SCT は拡張パックを作成します。この拡張パックは、AWS Lambda と Amazon Simple Email Service (Amazon SES) を使用します。AWS Lambda は、Amazon SES E メール送信サービスとやり取りするためのインターフェイスをユーザーに提供します。このインタラクションを設定するには、Lambda 関数の Amazon リソースネーム (ARN) を追加します。

新しいE メールアカウントの場合、次のコマンドを使用します。

```
do
$$
begin
PERFORM sysmail_add_account_sp (
```

```
    par_account_name :='your_account_name',
    par_email_address := 'your_account_email',
    par_display_name := 'your_account_display_name',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'ARN'
);
end;
$$ language plpgsql;
```

Lambda 関数の ARN を既存の E メールアカウントに追加するには、次のコマンドを使用します。

```
do
$$
begin
PERFORM sysmail_update_account_sp (
    par_account_name :='existind_account_name',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'ARN'
);
end;
$$ language plpgsql;
```

前述の例では、*ARN* は Lambda 関数の ARN です。

PostgreSQL での SQL Server データベースメールの動作をエミュレートするために、AWS SCT 拡張パックは次のテーブル、ビュー、およびプロシージャを使用します。

PostgreSQL で SQL サーバーデータベースメールをエミュレートするテーブル

SQL Server データベースメールをエミュレートするために、拡張パックは次のテーブルを使用します。

sysmail_account

E メールアカウントに関する情報を格納します。

sysmail_profile

ユーザープロファイルに関する情報を格納します。

sysmail_server

E メールサーバに関する情報を格納します。

sysmail_mail_items

メールメッセージのリストを保存します。

sysmail_attachments

E メール添付ごとに 1 つの行が含まれます。

sysmail_log

E メール送信に関するサービス情報を格納します。

sysmail_profile_account

ユーザープロファイルとE メールアカウントに関する情報を格納します。

PostgreSQL で SQL サーバーデータベースメールをエミュレートするビュー

SQL Server データベースメールをエミュレートするために、AWS SCT はPostgreSQL データベースに次のビューを作成して互換性を確保します。拡張パックではこれらのビューを使用しませんが、変換したコードでこれらのビューをクエリできます。

sysmail_allitems

すべてのメールのリストが含まれます。

sysmail_failedItems

送信できなかったメールのリストが含まれます。

sysmail_sentitems

送信済みメールのリストが含まれます。

sysmail_unsentims

まだ送信されていないメールのリストが含まれます。

sysmail_mail_attachments

添付ファイルのリストが含まれます。

PostgreSQL で SQL サーバーデータベースメールをエミュレートするプロシージャ

SQL Server データベースメールをエミュレートするために、拡張パックは次の手順を使用します。

sp_send_dbmail

指定した受信者に E メールを送信します。

sysmail_add_profile_sp

新しいユーザープロファイルを作成します。

sysmail_add_account_sp

簡易メール転送プロトコル (SMTP) 認証情報などの情報を保存する新しいメールアカウントを作成します。

sysmail_add_profileaccount_sp

指定したユーザープロファイルにメールアカウントを追加します。

sysmail_update_profile_sp

説明、名前など、ユーザープロファイルの属性を変更します。

sysmail_update_account_sp

既存のメールアカウントの情報を変更します。

sysmail_update_profile_account_sp

指定したユーザープロファイルのメールアカウント情報を更新します。

sysmail_delete_profile_account_sp

指定したユーザープロファイルからメールアカウントを削除します。

sysmail_delete_account_sp

E メールアカウントを削除します。

sysmail_delete_profile_sp

ユーザープロファイルを削除します。

sysmail_delete_mail_tems_sp

内部テーブルからメールを削除します。

sysmail_help_profile_sp

ユーザープロファイルに関する情報を表示します。

sysmail_help_account_sp

E メールアカウントに関する情報を表示します。

sysmail_help_profile_account_sp

ユーザープロファイルに関連付けられている電子メールアカウントに関する情報を表示します。

sysmail_dbmail_json

AWS Lambda 関数の JSON リクエストを生成する内部プロシージャ。

sysmail_verify_profile_sp、sysmail_verify_account_sp、sysmail_verify_addressparams_sp

設定を確認する内部プロシージャ。

sp_get_dbmail、sp_set_dbmail、sysmail_dbmail_xml

非推奨の内部プロシージャ。

PostgreSQL で SQL Server データベースメールをエミュレートするプロシージャの構文

拡張パック内の `aws_sqlserver_ext.sp_send_dbmail` プロシージャは、`msdb.dbo.sp_send_dbmail` プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_name varchar = NULL::character varying,
par_recipients text = NULL::text,
par_copy_recipients text = NULL::text,
par_blind_copy_recipients text = NULL::text,
par_subject varchar = NULL::character varying,
par_body text = NULL::text,
par_body_format varchar = NULL::character varying,
par_importance varchar = 'NORMAL'::character varying,
par_sensitivity varchar = 'NORMAL'::character varying,
par_file_attachments text = NULL::text,
par_query text = NULL::text,
par_execute_query_database varchar = NULL::character varying,
par_attach_query_result_as_file smallint = 0,
par_query_attachment_filename varchar = NULL::character varying,
par_query_result_header smallint = 1,
par_query_result_width integer = 256,
par_query_result_separator VARCHAR = ' '::character varying,
par_exclude_query_output smallint = 0,
par_append_query_error smallint = 0,
```

```
par_query_no_truncate smallint = 0,
par_query_result_no_padding smallint = 0,
out par_mailitem_id integer,
par_from_address text = NULL::text,
par_reply_to text = NULL::text,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_delete_mailitems_sp プロシージャは、msdb.dbo.sysmail_delete_mailitems_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_sent_before timestamp = NULL::timestamp without time zone,
par_sent_status varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_add_profile_sp プロシージャは、msdb.dbo.sysmail_add_profile_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_name varchar,
par_description varchar = NULL::character varying,
out par_profile_id integer,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_add_account_sp プロシージャは、msdb.dbo.sysmail_add_account_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_account_name varchar
par_email_address varchar
par_display_name varchar = NULL::character varying
par_replyto_address varchar = NULL::character varying
par_description varchar = NULL::character varying
par_mailserver_name varchar = NULL::character varying
par_mailserver_type varchar = 'SMTP'::bpchar
par_port integer = 25
par_username varchar = NULL::character varying
```

```
par_password varchar = NULL::character varying
par_use_default_credentials smallint = 0
par_enable_ssl smallint = 0
out par_account_id integer
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_add_profileaccount_sp プロシージャは、msdb.dbo.sysmail_add_profileaccount_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_sequence_number integer = NULL::integer,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_help_profile_sp プロシージャは、msdb.dbo.sysmail_help_profile_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_update_profile_sp プロシージャは、msdb.dbo.sysmail_update_profile_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_description varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_delete_profile_sp プロシージャは、msdb.dbo.sysmail_delete_profile_sp プロシージャをエミュレートします。SQL Server

データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_force_delete smallint = 1,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_help_account_sp プロシージャは、msdb.dbo.sysmail_help_account_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_update_account_sp プロシージャは、msdb.dbo.sysmail_update_account_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_email_address varchar = NULL::character varying,
par_display_name varchar = NULL::character varying,
par_replyto_address varchar = NULL::character varying,
par_description varchar = NULL::character varying,
par_mailserver_name varchar = NULL::character varying,
par_mailserver_type varchar = NULL::character varying,
par_port integer = NULL::integer,
par_username varchar = NULL::character varying,
par_password varchar = NULL::character varying,
par_use_default_credentials smallint = NULL::smallint,
par_enable_ssl smallint = NULL::smallint,
par_timeout integer = NULL::integer,
par_no_credential_change smallint = NULL::smallint,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_delete_account_sp プロシージャは、msdb.dbo.sysmail_delete_account_sp プロシージャをエミュレートします。SQL Server

データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_help_profileaccount_sp プロシージャは、msdb.dbo.sysmail_help_profileaccount_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_update_profileaccount_sp プロシージャは、msdb.dbo.sysmail_update_profileaccount_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_sequence_number integer = NULL::integer,
out returncode integer
```

拡張パック内の aws_sqlserver_ext.sysmail_delete_profileaccount_sp プロシージャは、msdb.dbo.sysmail_delete_profileaccount_sp プロシージャをエミュレートします。SQL Server データベースメールのソース手順の詳細については、『Microsoft 技術ドキュメント』を参照してください。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

PostgreSQL で SQL Server データベースメールをエミュレートするプロシージャを使用する場合の例

E メールを送信するには、次に示す aws_sqlserver_ext.sp_send_dbmail プロシージャに従います。

```
PERFORM sp_send_dbmail (
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Automated Success Message',
    par_body := 'The stored procedure finished'
);
```

次の例は、クエリ結果を E メールで送信する方法を示しています。

```
PERFORM sp_send_dbmail (
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Account with id = 1',
    par_query := 'SELECT COUNT(*)FROM Account WHERE id = 1'
);
```

次の例は、HTML コードを使用して E メールを送信する方法を示しています。

```
DECLARE var_tableHTML TEXT;
SET var_tableHTML := CONCAT(
    '<H1>Work Order Report</H1>',
    '<table border="1">',
    '<tr><th>Work Order ID</th><th>Product ID</th>',
    '<th>Name</th><th>Order Qty</th><th>Due Date</th>',
    '<th>Expected Revenue</th></tr>',
    '</table>'
);
PERFORM sp_send_dbmail (
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Work Order List',
    par_body := var_tableHTML,
    par_body_format := 'HTML'
);
```

E メールを削除するには、次に示す aws_sqlserver_ext.sysmail_delete_mailitems_sp の手順に従います。

```
DECLARE var_GETDATE datetime;
SET var_GETDATE = NOW();
PERFORM sysmail_delete_mailitems_sp (
    par_sent_before := var_GETDATE
);
```

次の例は、最も古い E メールを削除する方法を示しています。

```
PERFORM sysmail_delete_mailitems_sp (
    par_sent_before := '31.12.2015'
);
```

次の例は、送信できないすべての E メールを削除する方法を示しています。

```
PERFORM sysmail_delete_mailitems_sp (
    par_sent_status := 'failed'
);
```

新しいユーザープロファイルを作成するには、次に示す `aws_sqlserver_ext.sysmail_add_profile_sp` プロシージャに従います。

```
PERFORM sysmail_add_profile_sp (
    profile_name := 'Administrator',
    par_description := 'administrative mail'
);
```

次の例は、新しいプロファイルを作成し、一意なプロファイル ID を変数に保存する方法を示しています。

```
DECLARE var_profileId INT;
SELECT par_profile_id
    FROM sysmail_add_profile_sp (
        profile_name := 'Administrator',
        par_description := ' Profile used for administrative mail.')
    INTO var_profileId;

SELECT var_profileId;
```

新しいメールアカウントを作成するには、次に示す `aws_sqlserver_ext.sysmail_add_account_sp` プロシージャに従います。

```
PERFORM sysmail_add_account_sp (
    par_account_name :='Audit Account',
    par_email_address := 'dba@rusgl.info',
    par_display_name := 'Test Automated Mailer',
    par_description := 'Account for administrative e-mail.',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'arn:aws:lambda:us-west-2:555555555555:function:pg_v3'
);
```

ユーザープロファイルにメールアカウントを追加するには、次に示す aws_sqlserver_ext.sysmail_add_profileaccount_sp プロシージャに従います。

```
PERFORM sysmail_add_profileaccount_sp (
    par_account_name := 'Administrator',
    par_account_name := 'Audit Account',
    par_sequence_number := 1
);
```

PostgreSQL で SQL Server データベースメールをエミュレートするためのユースケースの例

ソースデータベースコードで SQL Server Database Mail を使用して電子メールを送信する場合は、AWS SCT 拡張パックを使用してこのコードを PostgreSQL に変換できます。

PostgreSQL データベースからE メールを送信するには

1. AWS Lambda 関数の作成と設定
2. AWS SCT 拡張パックを適用するには
3. 次に示す sysmail_add_profile_sp 関数を使用してユーザープロファイルを作成します。
4. 次に示す sysmail_add_account_sp 関数を使用してメールアカウントを作成します。
5. 次に示す sysmail_add_profileaccount_sp 関数を使用して、このメールアカウントをユーザープロファイルに追加します。

```
CREATE OR REPLACE FUNCTION aws_sqlserver_ext.
proc_dbmail_settings_msdb()
RETURNS void
AS
$BODY$
BEGIN
PERFORM aws_sqlserver_ext.sysmail_add_profile_sp(
    par_profile_name := 'Administrator',
```

```
    par_description := 'administrative mail'
);
PERFORM aws_sqlserver_ext.sysmail_add_account_sp(
    par_account_name := 'Audit Account',
    par_description := 'Account for administrative e-mail.',
    par_email_address := 'dba@rusgl.info',
    par_display_name := 'Test Automated Mailer',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'your_ARN'
);
PERFORM aws_sqlserver_ext.sysmail_add_profileaccount_sp(
    par_profile_name := 'Administrator',
    par_account_name := 'Audit Account',
    par_sequence_number := 1
);
END;
$BODY$
LANGUAGE plpgsql;
```

6. 次の sp_send_dbmail 関数を使用して E メールを送信します。

```
CREATE OR REPLACE FUNCTION aws_sqlserver_ext.
proc_dbmail_send_msdb()
RETURNS void
AS
$BODY$
BEGIN
PERFORM aws_sqlserver_ext.sp_send_dbmail(
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_body := 'The stored procedure finished',
    par_subject := 'Automated Success Message'
);
END;
$BODY$
LANGUAGE plpgsql;
```

すべてのユーザープロファイルに関する情報を表示するには、次に示す sysmail_help_profile_sp プロシージャに従います。

```
SELECT FROM aws_sqlserver_ext.sysmail_help_profile_sp();
```

次の例では、特定のユーザー プロファイルに関する情報を表示します。

```
select from aws_sqlserver_ext.sysmail_help_profile_sp(par_profile_id := 1);
select from aws_sqlserver_ext.sysmail_help_profile_sp(par_profile_name :=
 'Administrator');
```

すべてのメールアカウントに関する情報を表示するには、次に示す sysmail_help_account_sp プロシージャに従います。

```
select from aws_sqlserver_ext.sysmail_help_account_sp();
```

次の例では、特定のE メールアカウントに関する情報を表示します。

```
select from aws_sqlserver_ext.sysmail_help_account_sp(par_account_id := 1);
select from aws_sqlserver_ext.sysmail_help_account_sp(par_account_name := 'Audit
 Account');
```

ユーザープロファイルに関連付けられているすべてのメールアカウントに関する情報を表示するには、次に示す sysmail_help_profileaccount_sp プロシージャに従います。

```
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp();
```

次の例では、ID、プロファイル名、またはアカウント名でレコードをフィルタリングします。

```
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_id := 1);
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_id := 1,
 par_account_id := 1);
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_name :=
 'Administrator');
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_account_name := 'Audit
 Account');
```

ユーザープロファイル名または説明を変更するには、次に示す sysmail_update_profile_sp プロシージャに従います。

```
select aws_sqlserver_ext.sysmail_update_profile_sp(
    par_profile_id := 2,
    par_profile_name := 'New profile name'
);
```

メールアカウントの設定を変更するには、次に示す ysmail_update_account_sp プロシージャに従います。

```
select from aws_sqlserver_ext.sysmail_update_account_sp (
    par_account_name := 'Audit Account',
    par_mailserver_name := 'arn:aws:lambda:region:XXXXXXXXXXXX:function:func_test',
    par_mailserver_type := 'AWSLAMBDA'
);
```

SQL Server から Amazon RDS for SQL Server への変換

SQL Server スキーマとコードを Amazon RDS for SQL Server に移行する際は以下の点を考慮してください。

- AWS SCT は SQL Server エージェントを変換し、Amazon RDS for SQL Server DB インスタンスでスケジュール、アラート、およびジョブを提供できます。変換後、Amazon RDS for SQL Server DB インスタンスを SQL Server Reporting Service (SSRS)、SQL Server Analysis Services (SSAS)、および SQL Server Integration Services (SSIS) と使用できます。
- Amazon RDS は現在、SQL Server サービスブローカーまたは CREATE ENDPOINT コマンドを実行する必要がある追加の T-SQL エンドポイントをサポートしていません。
- Amazon RDS には、リンクされたサーバーのサポートに制限があります。リンクされたサーバーを使用する SQL Server アプリケーションコードを変換する場合、AWS SCT はアプリケーションコードを変換します。ただし、変換されたコードを実行する前に、リンクサーバーを使用するオブジェクトの動作を確認してください。
- 常にオンで使用されます。
- AWS SCT 評価レポートには、変換に関するサーバーのメトリクスが提供されています。SQL Server インスタンスに関するこれらのメトリクスには、次のようなものがあります。
  - データミラーリングを使用する。
  - SQL Server ログ配布を設定する。
  - フェイルオーバークラスターを使用する。
  - データベースメールを設定する。
  - 全文検索サービスを使用する。Amazon RDS for SQL Server の全文検索は制限されており、セマンティック検索はサポートされません。
  - Data Quality Service (DQS) をインストールする。Amazon RDS は DQS をサポートしていないため、SQL Server を Amazon EC2 インスタンスにインストールすることをお勧めします。

ターゲットとして RDS for SQL を使用する場合の権限

RDS for SQL Server に移行するには、データベースユーザーを作成し、各データベースに必要な権限を付与します。次のコード例を使用できます。

```
CREATE LOGIN user_name WITH PASSWORD 'your_password';

USE db_name
CREATE USER user_name FOR LOGIN user_name
GRANT VIEW DEFINITION TO user_name
GRANT VIEW DATABASE STATE TO user_name
GRANT CREATE SCHEMA TO user_name;
GRANT CREATE TABLE TO user_name;
GRANT CREATE VIEW TO user_name;
GRANT CREATE TYPE TO user_name;
GRANT CREATE DEFAULT TO user_name;
GRANT CREATE FUNCTION TO user_name;
GRANT CREATE PROCEDURE TO user_name;
GRANT CREATE ASSEMBLY TO user_name;
GRANT CREATE AGGREGATE TO user_name;
GRANT CREATE FULLTEXT CATALOG TO user_name;
GRANT CREATE SYNONYM TO user_name;
GRANT CREATE XML SCHEMA COLLECTION TO user_name;
```

上記の例の *user_name* は、使用するユーザー名に置き換えます。[*db_name*] をターゲットデータベースの名前に置き換えます。最後に、[*your_password*] を安全なパスワードに置き換えます。

AWS Schema Conversion Tool のデータウェアハウスソース

AWS SCT は、次のソースデータウェアハウスのスキーマを、サポートされているターゲットに変換できます。アクセス権限、接続、および AWS SCT がターゲットデータベースまたはデータウェアハウスで使用するために変換できるものについては、次の詳細を参照してください。

トピック

- AWS SCT のソースとしての Amazon Redshift の使用
- AWS SCT のソースとしての Azure Synapse Analytics の使用
- AWS SCT の BigQuery をソースとして使用する
- AWS SCT のソースとしての Greenplum データベースの使用
- AWS SCT のソースとしての Netezza の使用

- AWS SCT のソースとしての Oracle データウェアハウスの使用
- AWS SCT のソースとしての Snowflake の使用
- SQL Server データウェアハウスを AWS SCT のソースとしての使用
- AWS SCT ソースとしての Teradata の使用
- AWS SCT ソースとしての Vertica の使用

AWS SCT のソースとしての Amazon Redshift の使用

Amazon Redshift クラスターを最適化するには、AWS SCT を使用することができます。AWS SCT に、Amazon Redshift クラスターのディストリビューションキーとソートキーの選択に関する推奨事項が記載されています。Amazon Redshift 最適化プロジェクトは、ソースとターゲットが異なる Amazon Redshift クラスターを指す AWS SCT プロジェクトと見なすことができます。

ソースデータベースとしての Amazon Redshift の権限

Amazon Redshift をソースとして使用するには、次の権限が必要です。

- USAGE ON SCHEMA *<schema_name>*
- SELECT ON ALL TABLES IN SCHEMA *<schema_name>*
- SELECT ON PG_CATALOG.PG_STATISTIC
- SELECT ON SVV_TABLE_INFO
- SELECT ON TABLE STV_BLOCKLIST
- SELECT ON TABLE STV_TBL_PERM
- SELECT ON SYS_SERVERLESS_USAGE
- SELECT ON PG_DATABASE_INFO
- SELECT ON PG_STATISTIC

前述の例では、*<schema_name>* プレースホルダーをソースデータベースの名前に置き換えます。

ターゲットとしての Amazon Redshift に必要な権限については、「ターゲットとしての Amazon Redshift の許可」をご参照ください。

ソースとしての Amazon Redshift への接続

以下の手順を使用して、Amazon Redshift ソースデータベースを AWS Schema Conversion Tool に接続します。

Amazon Redshift ソースデータベースへの接続

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Amazon Redshift] を選択し、次に [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - Amazon Redshift ソースデータベースの接続情報を入力するには、以下の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
データベース	Amazon Redshift データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じ

パラメータ	アクション
	て再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー認証情報を確認する]: 信頼ストアを使用してサーバー証明書を確認するには、このオプションを選択します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。この場所を [グローバル設定] に追加すると、ここに表示されます。 Amazon Redshift の SSL サポートの詳細については、「接続のセキュリティオプションを設定する」を参照してください。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[Redshift ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Amazon Redshift 最適化設定

Amazon Redshift 最適化設定を編集するには、[設定] を選択しAWS SCT、[コンバージョン設定] を選択します。上のリストから [Amazon Redshift] を選択し、次に [Amazon Redshift — Amazon Redshift] を選択します。AWS SCT に、Amazon Redshift の最適化に使用可能なすべての設定が表示されます。

AWS SCT の Amazon Redshift 最適化設定には、以下のオプションが含まれます。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  テーブルの数が Amazon Redshift クラスターに保存できる数を超えている場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。
- 移行戦略を選択するには。

  AWS では、最適化プロジェクトのソースとターゲットとして異なるクラスターを使用することを推奨しています。Amazon Redshift 最適化プロセスを開始する前に、ソース Amazon Redshift ク

ラスターのコピーを作成します。このコピーにソースデータを含めることも、空のクラスターを作成することもできます。

[移行戦略] では、[コピーに移行] を選択して、ソースクラスターのデータをターゲットクラスターに含めます。

[移行戦略] では、[クリーンな状態に移行する] を選択し、最適化の提案を確認します。これらの提案を受け入れたら、ソースデータをターゲットクラスターに移行します。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

- 自動テーブル最適化を使用するには。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化にのみ使用するには、左側のペインで [最適化戦略] を選択します。次に [Amazon Redshift 自動テーブルチューニングを使用する] を選択し、[初期キー選択戦略] で [なし] を選択します。

- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。

  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
  - [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
  - [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
  - [統計ユーザーを選択] では、クエリ統計を分析するデータベースユーザーを選択します。

AWS SCT のソースとしての Azure Synapse Analytics の使用

AWS SCT を使用して、Azure Synapse Analytics からのスキーマ、コードオブジェクトおよびアプリケーションコードを Amazon Redshift に変換できます。

ソースデータベースとしての Azure Synapse Analytics の権限

Azure Synapse Analytics データ ウェアハウスをソースとして使用するには、次の権限が必要です。

- VIEW DEFINITION
- VIEW DATABASE STATE

スキーマを変換するデータベースごとに権限を適用します。

ソースとしての Azure Synapse Analytics への接続

以下の手順を使用して、AWS Schema Conversion Tool を含む Azure Synapse Analytics データウェアハウスに接続します。

Azure Synapse Analytics データ ウェアハウスにソースとして接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Azure Synapse Analytics] を選択し、次に[Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - Azure Synapse Analytics データウェアハウスの接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームサービス (DNS) 名または IP アドレスを入力します。
SQL プール	Azure SQL プールの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワ

パラメータ	アクション
	ードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー証明書を信頼する]: サーバー証明書を信頼するには、このオプションを選択します。 • [信頼ストア]: [グローバル設定] でセットアップした信頼ストア。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Azure Synapse Analytics から Amazon Redshift への変換設定

Azure Synapse Analytics から Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Azure Synapse] を選択し、次に [Azure Synapse – Amazon Redshift] を選択します。AWS SCT に、Azure Synapse Analytics から Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT の Azure Synapse Analytics から Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

[変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。

- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。

- ソーステーブルのパーティションを Amazon Redshift の別のテーブルに移行するには。そのためには、[UNION ALL ビューを使用する] を選択し、AWS SCT が 1 つのソーステーブルに対して作成できるターゲットテーブルの最大数を入力します。

  Amazon Redshift は、テーブルのパーティションをサポートしていません。この動作をエミュレートしてクエリをより速く実行するために、AWS SCT は、ソーステーブルの各パーティションを Amazon Redshift の個別のテーブルに移行できます。次に、AWS SCT はこれらすべてのテーブルのデータを含むビューを作成します。

  AWS SCT は、ソーステーブルのパーティションの数を自動的に決定します。ソーステーブルパーティショニングのタイプによっては、この数は Amazon Redshift クラスターに適用できるテーブルのクォータを超える場合があります。このクォータに達しないようにするには、AWS SCT が 1 つのソーステーブルのパーティションに対して作成できるターゲットテーブルの最大数を入力し

ます。デフォルトのオプションは 368 テーブルで、これは 1 年 366 日のパーティションと、NO RANGE および UNKNOWN パーティションの 2 つのテーブルを表します。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

Azure Synapse Analytics から Amazon Redshift への変換最適化設定

Azure Synapse Analytics から Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Azure Synapse] を選択し、次に [Azure Synapse — Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT で、Azure Synapse Analytics から Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT の Azure Synapse Analytics から Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。

- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。

- メタデータを使用し、統計情報は無視する
- メタデータを無視し、統計情報を使用する
- メタデータと統計情報を使用する

選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

- 戦略の詳細を設定するには。

各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

- [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
- [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
- [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
- [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT の BigQuery をソースとして使用する

AWS SCT を使用して、Teradata からのスキーマ、コードオブジェクトおよびアプリケーションコードを BigQuery から Amazon Redshift に変換できます。

BigQuery をソースとする場合の権限

AWS SCT で BigQuery データウェアハウスをソースとして使用するには、サービスアカウントを作成します。Google Cloud では、アプリケーションはサービスアカウントを使用して承認された API 呼び出しを行います。サービスアカウントはユーザーアカウントとは異なります。詳細については、Google Cloud Identity and Access Management ドキュメントの「サービス アカウント」を参照してください。

サービスアカウントには必ず以下のロールを付与してください。

- `BigQuery Admin`
- `Storage Admin`

`BigQuery Admin` ロールは、プロジェクト内のすべてのリソースを管理する権限を付与します。AWS SCT は、このロールを使用して移行プロジェクトに BigQuery メタデータを読み込みます。

`Storage Admin` ロールにより、データオブジェクトとバケットを完全に制御できます。このロールは `Cloud Storage` の下にあります。AWS SCTは このロールを使用して BigQuery からデータを抽出し、それを Amazon Redshift にロードします。

サービスアカウントキーファイルを作成するには

1. https://console.cloud.google.com/ で Google Cloud 管理コンソールにログインします。
2. BigQuery API ページで [有効にする] を選択します。API が [有効] になっている場合は、このステップをスキップしてください。
3. [サービスアカウント] ページで、プロジェクトを選択し、[サービスアカウントの作成] を選択します。
4. [サービスアカウント詳細] ページで、[サービスアカウント名] にわかりやすい値を入力します。[作成して続ける] を選択します。[このサービスアカウントにプロジェクトへのアクセス権を付与] ページが開きます。
5. [ロールの選択] で [BigQuery] を選択し、次に [BigQuery 管理者] を選択します。
6. [別のロールを追加] を選択します。[ロールの選択] で [クラウドストレージ] を選択し、[ストレージ管理者] を選択します。
7. [続ける] を選択し、次に [保存] を選択します。
8. [サービスアカウント] ページで、作成したサービスアカウントを選択します。
9. [キー] を選択し、[キーの追加] で [新しいキーの作成] を選択します。

10. [JSON]、[作成] の順に選択します。プライベートキーを保存するフォルダを選択するか、ブラウザのダウンロード用のデフォルトフォルダを選択します。

BigQuery データウェアハウスからデータを抽出するために、AWS SCT は Google Cloud Storage バケットフォルダを使用します。データ移行を開始する前に、このバケットを作成してください。Google Cloud Storage バケットフォルダへのパスを [ローカルタスクの作成] ダイアログボックスに入力します。詳細については、「AWS SCT タスクの作成、実行、監視」を参照してください。

BigQuery へソースとして接続

以下の手順を使用して、ソースデータベースを AWS Schema Conversion Tool に接続します。

BigQuery ソースデータウェアハウスに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [BigQuery] を選択し、[次へ] を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] に BigQuery プロジェクトの名前を入力します。AWS SCT で、この名前が左側のパネルのツリーに表示されます。
4. [キーパス] にサービスアカウントのキーファイルへのパスを入力します。このファイルの作成の詳細については、「BigQuery をソースとする場合の権限」を参照してください。
5. [接続のテスト] を選択して、AWS SCT をソースデータベースに正常に接続できることを確認します。
6. [接続] を選択して、ソース BigQuery プロジェクトに接続します。

BigQuery を AWS SCT のソースとして使用する場合の制限

BigQuery を AWS SCT のソースとして使用する場合は、以下の制限が適用されます。

- AWS SCT では、分析関数でのサブクエリの変換はサポートされていません。
- AWS SCT を使用して BigQuery `SELECT AS STRUCT` と `SELECT AS VALUE` ステートメントを変換することははできません。
- AWS SCT では、次のタイプの関数の変換はサポートされていません。
  - 近似値

- ビット
- デバッグ
- 横串検索
- 地域別
- ハッシュ
- 数学的
- 正味
- 統計集計
- UUID

- AWS SCT では、文字列関数の変換は限定的なサポートされているだけです。
- AWS SCT では、UNNEST 演算子の変換はサポートされていません。
- AWS SCT では相関結合操作を変換できません。
- AWS SCT、QUALIFY、WINDOW、LIMIT、OFFSET 句の変換はサポートされていません。
- AWS SCT を使用して再帰的な共通テーブル式を変換することはできません。
- AWS SCT では、VALUES句内にサブクエリを含む INSERT ステートメントの変換はサポートされていません。
- AWS SCT では、ネストされたフィールドや繰り返しレコードの UPDATE ステートメントの変換はサポートされていません。
- AWS SCT を STRUCT および ARRAYデータ型の変換には使用することはできません。

BigQuery から Amazon Redshift への変換設定

BigQuery から Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Google BigQuery] を選択し、次に [Google BigQuery – Amazon Redshift] を選択します。AWS SCT に、BigQuery から Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT での BigQuery から Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

[変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。

- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

BigQuery から Amazon Redshift への変換最適化設定

BigQuery から Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Google BigQuery] を選択し、次に [Google BigQuery — Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に BigQuery から Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT での BigQuery から Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。
- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。
  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

  選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

  [小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。
- 戦略の詳細を設定するには。

各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

- [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
- [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
- [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
- [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT のソースとしての Greenplum データベースの使用

AWS SCT を使用して、Greenplum データベースからのスキーマ、コードオブジェクト、およびアプリケーションコードを Amazon Redshift に変換できます。

ソースとしての Greenplum データベースの権限

Greenplum データベースをソースとして使用するのに必要な権限を以下に示します。

- CONNECT ON DATABASE *<database_name>*
- USAGE ON SCHEMA *<schema_name>*
- SELECT ON *<schema_name>.<table_name>*
- SELECT ON SEQUENCE *<schema_name>.<sequence_name>*

上記の例で、次のプレースホルダを置き換えます。

- *database_name* をソースデータベースの名前に置き換えます。
- *schema_name* をソーススキーマの名前に置き換えます。
- *table_name* をテーブルの名前に置き換えます。
- *sequence_name* をシーケンスの名前に置き換えます。

ソースとしての Greenplum データベースへの接続

以下の手順を使用して、Greenplum ソースデータベースを AWS SCT に接続します。

Greenplum ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [SAP ASE] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - Greenplum ソースデータベースの認証情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
データベース	Greenplum データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。

パラメータ	アクション
	AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー認証情報を確認する]: 信頼ストアを使用してサーバー証明書を確認するには、このオプションを選択します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[Greenplum データベースドライバパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Greenplum から Amazon Redshift への変換設定

Greenplum から Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、次に [変換設定] を選択します。上のリストから [Greenplum] を選択し、次に [Greenplum – Amazon Redshift] を選択します。AWS SCT に、Greenplum から Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT の GreenplumからAmazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。
- ソーステーブルのパーティションを Amazon Redshift の別のテーブルに移行するには。そのためには、[UNION ALL ビューを使用する] を選択し、AWS SCT が 1 つのソーステーブルに対して作成できるターゲットテーブルの最大数を入力します。

Amazon Redshift は、テーブルのパーティションをサポートしていません。この動作をエミュレートしてクエリをより速く実行するために、AWS SCT は、ソーステーブルの各パーティションを Amazon Redshift の個別のテーブルに移行できます。次に、AWS SCT はこれらすべてのテーブルのデータを含むビューを作成します。

AWS SCT は、ソーステーブルのパーティションの数を自動的に決定します。ソーステーブルパーティショニングのタイプによっては、この数は Amazon Redshift クラスターに適用できるテーブルのクォータを超える場合があります。このクォータに達しないようにするには、AWS SCT が 1 つのソーステーブルのパーティションに対して作成できるターゲットテーブルの最大数を入力します。デフォルトのオプションは 368 テーブルで、これは 1 年 366 日のパーティションと、NO RANGE および UNKNOWN パーティションの 2 つのテーブルを表します。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

Greenplum から Amazon Redshift への変換最適化設定

Greenplum から Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、次に [変換設定] を選択します。上のリストから [Greenplum] を選択し、次に [Greenplum – Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に Greenplumから Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT の Greenplum から Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。

- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。

  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

  選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

  [小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
  - [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
  - [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
  - [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT のソースとしての Netezza の使用

AWS SCT を使用して、Netezza からのスキーマ、コードオブジェクトおよびアプリケーションコードを Amazon Redshift に変換できます。

ソースとしての Netezza の権限

ソースとして Netezza に必要な特権を以下に示します。

- SELECT ON SYSTEM.DEFINITION_SCHEMA.SYSTEM VIEW
- SELECT ON SYSTEM.DEFINITION_SCHEMA.SYSTEM TABLE
- SELECT ON SYSTEM.DEFINITION_SCHEMA.MANAGEMENT TABLE
- LIST ON *<database_name>*
- LIST ON *<schema_name>*
- LIST ON *<database_name>*.ALL.TABLE
- LIST ON *<database_name>*.ALL.EXTERNAL TABLE
- LIST ON *<database_name>*.ALL.VIEW
- LIST ON *<database_name>*.ALL.MATERIALIZED VIEW
- LIST ON *<database_name>*.ALL.PROCEDURE
- LIST ON *<database_name>*.ALL.SEQUENCE
- LIST ON *<database_name>*.ALL.FUNCTION
- LIST ON *<database_name>*.ALL.AGGREGATE

上記の例で、次のプレースホルダを置き換えます。

- *database_name* をソースデータベースの名前に置き換えます。
- *schema_name* をソーススキーマの名前に置き換えます。

AWS SCTでは、次のシステムテーブルとビューへのアクセスが必要です。前述のリストにある `system.definition_schema.system view` および `system.definition_schema.system tables` へのアクセス権を付与する代わりに、これらのオブジェクトへのアクセス権を付与できます。

- select on system.definition_schema._t_aggregate
- select on system.definition_schema._t_class
- select on system.definition_schema._t_constraint
- select on system.definition_schema._t_const_relattr
- select on system.definition_schema._t_database
- select on system.definition_schema._t_grpobj_priv
- select on system.definition_schema._t_grpusr
- select on system.definition_schema._t_hist_config
- select on system.definition_schema._t_object
- select on system.definition_schema._t_object_classes
- select on system.definition_schema._t_proc
- select on system.definition_schema._t_type
- select on system.definition_schema._t_user
- select on system.definition_schema._t_usrobj_priv
- select on system.definition_schema._vt_sequence
- select on system.definition_schema._v_aggregate
- select on system.definition_schema._v_constraint_depends
- select on system.definition_schema._v_database
- select on system.definition_schema._v_datatype
- select on system.definition_schema._v_dslice
- select on system.definition_schema._v_function
- select on system.definition_schema._v_group
- select on system.definition_schema._v_obj_relation
- select on system.definition_schema._v_obj_relation_xdb
- select on system.definition_schema._v_procedure
- select on system.definition_schema._v_relation_column
- select on system.definition_schema._v_relation_keydata

- select on system.definition_schema._v_relobjclasses
- select on system.definition_schema._v_schema_xdb
- select on system.definition_schema._v_sequence
- select on system.definition_schema._v_synonym
- select on system.definition_schema._v_system_info
- select on system.definition_schema._v_sys_constraint
- select on system.definition_schema._v_sys_object_dslice_info
- select on system.definition_schema._v_sys_user
- select on system.definition_schema._v_table
- select on system.definition_schema._v_table_constraint
- select on system.definition_schema._v_table_dist_map
- select on system.definition_schema._v_table_organize_column
- select on system.definition_schema._v_table_storage_stat
- select on system.definition_schema._v_user
- select on system.definition_schema._v_view
- select on system.information_schema._v_relation_column
- select on system.information_schema._v_table
- select on $hist_column_access_*

ソースとしての Netezza への接続

以下の手順を使用して、Netezza ソースデータベースを AWS Schema Conversion Tool に接続します。

Netezza ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Netezza] [Next] (次へ) の順に選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

- Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。

  1. [AWS シークレット] で、シークレットの名前を選択します。
  2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

  Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。

- Netezza ソースデータベースの接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。

パラメータ	アクション
[Netezza のドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

継続的なデータ複製の設定

Netezza データベーススキーマを変換して Amazon Redshift データベースに適用したら、データ抽出エージェントを使用して AWS SCT データを移行できます。エージェントはデータを抽出し、Amazon S3 バケットにアップロードします。その後、AWS SCT を使用して Amazon S3 から Amazon Redshift にデータをコピーできます。

移行プロセス中にソースデータベースのデータが変更された場合は、AWS SCT データ抽出エージェントで進行中の変更をキャプチャできます。その後、最初のデータ移行を完了した後に、これらの継続的な変更をターゲットデータベースに複製できます。このプロセスは継続的レプリケーションまたは変更データキャプチャ (CDC) と呼ばれます。

Netezza から Amazon Redshift への移行のための継続的なデータレプリケーションを設定するには

1. ソースデータベースに、履歴データベースを作成します。Netezza コマンドラインインターフェイス (CLI) で次のコード例を使用できます。

```
nzhistcreatedb -d history_database_name -t query -v 1 -u load_user -o histdb_owner
 -p your_password
```

前述の例では、*history_database_name* を設定データベースの名前に置き換えます。次に、*load_user* を、履歴データをデータベースにロードするように定義したユーザーの名前に置き換えます。次に、*histdb_owner* を、履歴データベースの所有者として定義したユーザー

の名前に置き換えます。このユーザーをすでに作成し、CREATE DATABASE 権限を付与していることを確認してください。最後に、[*your_password*] を安全なパスワードに置き換えます。

2. 履歴ログを設定します。これを行うには、以下のコード例を使用します。

```
CREATE HISTORY CONFIGURATION history_configuration_name HISTTYPE QUERY
    DATABASE history_database_name USER load_user PASSWORD your_password COLLECT
 PLAN, COLUMN
    LOADINTERVAL 1 LOADMINTHRESHOLD 0 LOADMAXTHRESHOLD 0 STORAGELIMIT 25
    LOADRETRY 2 VERSION 1;
```

前の例では、*history_configuration_name* と *history_database_name* を、自分の履歴設定と履歴データベースの名前に置き換えます。次に、*load_user* を、履歴データをデータベースにロードするように定義したユーザーの名前に置き換えます。次に、*your_password* を安全なパスワードに置き換えます。

3. 履歴データベース内のすべてのテーブルに読み取り権限を付与します。次のコード例を使用して SELECT 権限を付与できます。

```
GRANT SELECT ON history_database_name.ALL.TABLE TO your_user;
```

前述の例では、*history_database_name* を設定データベースの名前に置き換えます。次に、*your_user* を Netezza データベースを操作するための最小限の権限しか持たないユーザーの名前に置き換えます。AWS SCT では、このデータベースユーザーの認証情報を使用します。

4. ソーススキーマ内の各テーブルの統計情報を収集して、列のカーディナリティに関する情報を取得します。以下のコマンドを使用して、履歴データベースの統計を生成できます。

```
GENERATE STATISTICS on "schema_name"."table_name";
```

前の例では、*schema_name* と *table_name* をデータベーススキーマとテーブルの名前に置き換えてください。

5. 次のクエリを実行して、前提条件を完了していることを確認します。

```
SELECT COUNT(*)
 FROM history_database_name.history_schema_name."$hist_column_access_N";
```

前の例では、*history_database_name* と *history_schema_name* を自分の履歴データベースとスキーマの名前に置き換えます。次に、*N* を履歴データベースのバージョン番号に置き換え

ます。履歴データベースのバージョンの詳細については、「IBM Netezza のドキュメント」を参照してください。

6. データ抽出エージェントをインストールします。詳細については、「抽出エージェントをインストールする」を参照してください。

   すべてのエクストラクターインスタンスの `settings.properties` ファイル内の `{working.folder}` パラメーターが同じフォルダを指していることを確認してください。この場合、エクストラクターは CDC セッションを調整し、すべてのサブタスクに 1 つのトランザクションポイントを使用できます。

7. データ抽出エージェントを登録します。詳細については、「抽出エージェントをに登録する AWS Schema Conversion Tool」を参照してください。

8. CDC タスクを作成します。詳細については、「 AWS SCT タスクの作成、実行、監視」を参照してください。

   a. AWS SCT でプロジェクトを開きます。左のペインで、ソーステーブルを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[ローカルタスクの作成] を選択します。

   b. [タスク名] に、データ移行タスクのわかりやすい名前を入力します。

   c. [移行モード] には、[抽出、アップロード、コピー] を選択します。

   d. [同期の有効化] を選択します。

   e. [CDC 設定] タブを選択し、CDC セッションの範囲とスケジュールを定義します。

   f. [タスクのテスト] を選択して、作業フォルダ、Amazon S3 バケット、および Amazon Redshift データウェアハウスに接続できることを確認します。

   g. [作成] を選択して、タスクを作成します。

   h. [タスク] タブを選択し、一覧からタスクを選択して [開始] を選択します。

9. AWS SCT タスクはターゲットデータベース上のトランザクションの一貫性を維持します。データ抽出エージェントは、ソースからのトランザクションをトランザクション ID 順に複製します。

   移行セッションのいずれかを停止したり、失敗したりすると、CDC 処理も停止します。

Netezza から Amazon Redshift への変換設定

Netezza から Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Netezza] を選択し、次に [Netezza – Amazon Redshift] を選択

します。AWS SCT に、Netezza から Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT での Netezza から Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。
- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

Netezza から Amazon Redshift への変換最適化設定

Netezza から Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Netezza] を選択し、次に [Netezza – Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に Netezzaから Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT での Netezza から Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。
- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。

  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

  選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
  - [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
  - [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
  - [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

  また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT のソースとしての Oracle データウェアハウスの使用

AWS SCT を使用して、スキーマ、コード・オブジェクトおよびアプリケーション・コードを Oracle データウェアハウスから Amazon Redshift または Amazon Redshift に変換し、AWS Glue を組み合わせて使用できます。

ソースとしての Oracle データウェアハウスの権限

Oracle データウェアハウスをソースとして使用するには、次の権限が必要です。

- 接続
- select_catalog_role
- ディクショナリを選択します

ソースとしての Oracle データウェアハウスへの接続

以下の手順を使用して、Oracle データウェアハウスソースデータベースを AWS Schema Conversion Tool に接続します。

Oracle データウェアハウスソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Oracle] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - Oracle ソースデータウェアハウス接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション

タイプ	データベースへの接続タイプを選択します。選択したタイプに応じて、以下の追加情報を提供します。
	• SID
	  • [サーバー名]: ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
	  • Server port: ソースデータベースサーバーへの接続に使用するポート。

パラメータ	アクション

	• Oracle SID: Oracle System ID (SID)。Oracle SID を見つけるには、Oracle データベースに対して以下のクエリを発行します。
	
	`SELECT sys_context('userenv','instance_name') AS SID FROM dual;`
	• サービス名
	  • Server name: ソースデータベースサーバーの DNS 名または IP アドレス。
	  • Server port: ソースデータベースサーバーへの接続に使用するポート。
	  • Service Name: 接続先の Oracle サービスの名前。
	• [TNS エイリアス]
	  • TNS file path: Transparent Network Substrate (TNS) 名接続情報を含むファイルへのパス。
	  • TNS file path: ソースデータベースへの接続に使用するこのファイルからの TNS エイリアス。
	• [TNS 接続識別子]
	  • [TNS 識別子]: 登録された TNS 接続情報の識別子。

User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。

パラメータ	アクション
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [SSL A認証]: 接続に SSL 認証を使用するには、このオプションを選択します。 • [信頼ストア]: 証明書を保存している信頼ストアの場所。 • [キーストア]: プライベートキーと証明書を保存しているキーストアの場所。この値は、[SSL 認証] を選択した場合は必須ですが、それ以外の場合はオプションです。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[Oracle ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Oracle データウェアハウスから Amazon Redshift への変換設定

Oracle データウェアハウスから Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、次に [変換設定] を選択します。上のリストから [Oracle] を選択し、次に [Oracle – Amazon Redshift] を選択します。AWS SCT に、Oracle データウェアハウスから Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT での Oracle データウェアハウスから Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。

- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。

- ソーステーブルのパーティションを Amazon Redshift の別のテーブルに移行するには。そのためには、[UNION ALL ビューを使用する] を選択し、AWS SCT が 1 つのソーステーブルに対して作成できるターゲットテーブルの最大数を入力します。

  Amazon Redshift は、テーブルのパーティションをサポートしていません。この動作をエミュレートしてクエリをより速く実行するために、AWS SCT は、ソーステーブルの各パーティションを Amazon Redshift の個別のテーブルに移行できます。次に、AWS SCT はこれらすべてのテーブルのデータを含むビューを作成します。

AWS SCT は、ソーステーブルのパーティションの数を自動的に決定します。ソーステーブルパーティショニングのタイプによっては、この数は Amazon Redshift クラスターに適用できるテーブルのクォータを超える場合があります。このクォータに達しないようにするには、AWS SCT が 1 つのソーステーブルのパーティションに対して作成できるターゲットテーブルの最大数を入力します。デフォルトのオプションは 368 テーブルで、これは 1 年 366 日のパーティションと、NO RANGE および UNKNOWN パーティションの 2 つのテーブルを表します。

- Amazon Redshift がサポートしていない日時フォーマット要素を使用して、TO_CHAR、TO_DATE、TO_NUMBER などのデータ型フォーマット関数を変換するには。デフォルトでは、AWS SCT は拡張パック関数を使用して、変換されたコード内のサポートされていないフォーマット要素の使用をエミュレートします。

  Oracle の日時フォーマットモデルには、Amazon Redshift の日時フォーマット文字列と比較してより多くの要素が含まれています。ソースコードに Amazon Redshift がサポートする日時形式の要素のみが含まれている場合、変換されたコードに拡張パック関数は必要ありません。変換後のコードで拡張パック関数を使用しないようにするには、Oracle コードで使用する [日付型フォーマット要素が Amazon Redshift の日時フォーマット文字列に似ている] ものを選択してください。この場合、変換されたコードの処理が速くなります。

  Oracle の数値フォーマットモデルには、Amazon Redshift の数値フォーマット文字列よりも多くの要素が含まれています。ソースコードに Amazon Redshift がサポートする数値形式の要素のみが含まれている場合、変換されたコードに拡張パック関数は必要ありません。変換後のコードで拡張パック関数を使用しないようにするには、[Oracle コードで使用する数値形式の要素は Amazon Redshift の数値形式の文字列と類似している] を選択します。この場合、変換されたコードの処理が速くなります。

- Oracle LEAD および LAG 分析関数を変換します。デフォルトでは、AWS SCT は LEAD および LAG 関数ごとにアクション項目が生成されます。

  ソースコードでこれらの関数でオフセットのデフォルト値を使用していない場合、AWS SCT は NVL 関数でこれらの関数の使用をエミュレートできます。そのためには、[NVL 関数を使用して Oracle LEAD 関数と LAG 関数の動作をエミュレートする] を選択します。

- Amazon Redshift クラスターのプライマリキーとユニークキーの動作をエミュレートするには、[プライマリキーとユニークキーの動作をエミュレートする] を選択します。

  Amazon Redshift はユニークキーやプライマリキーを強制せず、情報提供のみを目的として使用します。これらの制約をコードで使用する場合は、AWS SCT が変換後のコードでもその動作をエミュレートすることを確認してください。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

Oracle データウェアハウスから Amazon Redshift への変換最適化設定

Oracle データウェアハウスから Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、次に [変換設定] を選択します。上のリストから [Oracle] を選択し、次に [Oracle – Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に Oracle データウェアハウスから Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT での Oracle データウェアハウスから Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。
- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。
  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
  - [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
  - [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
  - [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

  また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT のソースとしての Snowflake の使用

AWS SCT を使用して、Snowflake からのスキーマ、コードオブジェクトおよびアプリケーションコードを Amazon Redshift に変換できます。

ソースデータベースとしての Snowflake の権限

権限を持つロールを作成し、SECURITYADMIN ロールと SECURITYADMIN セッションコンテキストを使用して、このロールにユーザーの名前を付与することができます。

次の例では、最小限の権限を作成し、min_privs ユーザーに付与しています。

```
create role role_name;
grant role role_name to role sysadmin;
grant usage on database db_name to role role_name;
grant usage on schema db_name.schema_name to role role_name;
grant usage on warehouse datawarehouse_name to role role_name;
grant monitor on database db_name to role role_name;
grant monitor on warehouse datawarehouse_name to role role_name;
grant select on all tables in schema db_name.schema_name to role role_name;
grant select on future tables in schema db_name.schema_name to role role_name;
grant select on all views in schema db_name.schema_name to role role_name;
grant select on future views in schema db_name.schema_name to role role_name;
grant select on all external tables in schema db_name.schema_name to role role_name;
grant select on future external tables in schema db_name.schema_name to role role_name;
grant usage on all sequences in schema db_name.schema_name to role role_name;
grant usage on future sequences in schema db_name.schema_name to role role_name;
grant usage on all functions in schema db_name.schema_name to role role_name;
grant usage on future functions in schema db_name.schema_name to role role_name;
grant usage on all procedures in schema db_name.schema_name to role role_name;
grant usage on future procedures in schema db_name.schema_name to role role_name;
create user min_privs password='real_user_password'
DEFAULT_ROLE = role_name DEFAULT_WAREHOUSE = 'datawarehouse_name';
grant role role_name to user min_privs;
```

上記の例で、次のプレースホルダを置き換えます。

- *role_name* を読み取り専用の権限を持つロールの名前に置き換えます。
- *db_name* をソースデータベースの名前に置き換えます。
- *schema_name* をソーススキーマの名前に置き換えます。
- *datawarehousename* を必要となるデータウェアハウスの名前に置き換えます。
- *min_privs* を最小限の権限を持つユーザーの名前に置き換えます。

DEFAULT_ROLE および DEFAULT_WAREHOUSE パラメータでは大文字と小文字が区別されます。

Amazon S3 へのセキュアアクセスの設定

Amazon S3 バケットにセキュリティおよびアクセス管理ポリシーを設けることで、Snowflake は S3 バケットにアクセスし、データの読み取りおよび書き込みができるようになります。Snowflake STORAGE INTEGRATION オブジェクトタイプを使用して、プライベート Amazon S3 バケットへの安全なアクセスを設定できます。Snowflake ストレージ統合オブジェクトは、認証責任を Snowflake の ID およびアクセス管理エンティティに委任します。

詳細については、Snowflake ドキュメントにある Amazon S3 にアクセスするための Snowflake ストレージ統合の設定を参照してください。

ソースとしての Snowflake への接続

以下の手順を使用して、 ソースデータベースを AWS Schema Conversion Tool に接続します。

Snowflake ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Snowflake] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - Snowflake ソースデータウェアハウス接続情報を手動で入力するには、以下の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
データベース	Snowflake データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。

パラメータ	アクション
	AWS SCT は、明示的にリクエストした場合にのみ、暗号形式でパスワードを保存します。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [プライベートキーパス]:プライベートキーの場所。 • [パスフレーズ]:プライベートキーのパスフレーズ。 Snowflake の SSL サポートの詳細については、「接続のセキュリティオプションを設定する」を参照してください。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションを設定すると、データベースパスワードを保存できます。これにより、パスワードを入力しなくてもデータベースにすばやく接続できるようになります。
Snowflake ドライバパス	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Snowflake をソースとして使用する場合の制限

Snowflake を AWS SCT のソースとして使用する場合の制限は次のとおりです。

- オブジェクト識別子は、オブジェクトタイプと親オブジェクトのコンテキスト内で一意である必要があります。

  データベース

  スキーマ識別子は、データベース内で一意である必要があります。

  スキーマ

  テーブルやビューなどのオブジェクト識別子は、スキーマ内で一意である必要があります。

  テーブル/ビュー

  列識別子は、テーブル内で一意である必要があります。

- 大規模および xlarge クラスターノードタイプのテーブルの最大数は 9,900 です。8xlarge クラスターノードタイプのテーブルの最大数は 100,000 です。制限には、ユーザー定義の一時テーブルと、クエリの処理またはシステムメンテナンス中に Amazon Redshift によって作成された一時テーブルの両方が含まれます。詳細については、Amazon Redshift クラスター管理ガイドの Amazon Redshift クォータを参照してください。
- ストアドプロシージャの場合、入出力引数の最大数は 32 です。

Snowflake のソースデータ型

次に、AWS SCT を使用する場合にサポートされる Snowflake ソースデータ型と、Amazon Redshift ターゲットへのデフォルトのマッピングを示します。

Snowflake のデータ型	Amazon Redshift のデータ型
NUMBER	NUMERIC(38)
NUMBER(p)	If p is =< 4, then SMALLINT If p is => 5 and =< 9, then INTEGER If p is => 10 and =< 18, then BIGINT If p is => 19 then NUMERIC(p)
NUMBER(p,0)	If p is =< 4, then SMALLINT If p is => 5 and =< 9, then INTEGER

Snowflake のデータ型	Amazon Redshift のデータ型
	If p is => 10 and =< 18, then BIGINT If p is => 19 then: NUMERIC(p,0)
NUMBER(p,s)	If p is => 1 and =< 38, and if s is => 1 and =< 37, then NUMERIC(p,s)
FLOAT	FLOAT
TEXT 16,777,216 バイトまでの Unicode 文字。1 文字あたり最大 4 バイト。	VARCHAR(MAX)
TEXT(p) 65,535 バイトまでの Unicode 文字。1 文字あたり最大 4 バイト。	If p is =< 65,535 then, VARCHAR(p)
TEXT(p) 16,777,216 バイトまでの Unicode 文字。1 文字あたり最大 4 バイト。	If p is => 65,535 and =< 16,777,216 then, VARCHAR(MAX)
BINARY 8 388,608 バイトまでのシングルバイト文字、1 文字あたり 1 バイト。	VARCHAR(MAX)
BINARY(p) 65,535 バイトまでのシングルバイト文字、1 文字あたり 1 バイト。	VARCHAR(p)
BINARY(p) 8 388,608 バイトまでのシングルバイト文字、1 文字あたり 1 バイト。	VARCHAR(MAX)

Snowflake のデータ型	Amazon Redshift のデータ型
BOOLEAN	BOOLEAN
DATE	DATE
TIME 00:00:00 から 23:59:59.999999999 の間の時間値。	VARCHAR(18)
TIME(f) 00:00:00 から 23:59:59.9(f) の間の時間値。	VARCHAR(n) – 9 + dt-attr-1
TIMESTAMP_NTZ	TIMESTAMP
TIMESTAMP_TZ	TIMESTAMPTZ

Snowflake から Amazon Redshift への変換設定

Snowflake から Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Snowflake] を選択し、次に [Snowflake – Amazon Redshift] を選択します。AWS SCT に、Snowflake から Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT での Snowflake から Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。

- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

スノーフレークから Amazon Redshift への変換最適化設定

Snowflake から Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Snowflake] を選択し、次に [Oracle – Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に Snowflake から Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT での Snowflake から Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。
- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。
  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

  選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

  [小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。
- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。
  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
  - [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
  - [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
  - [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

SQL Server データウェアハウスを AWS SCT のソースとしての使用

AWS SCT を使用して、スキーマ、コード・オブジェクトおよびアプリケーション・コードを Microsoft SQL Server DW から Amazon Redshift または Amazon Redshift に変換し、AWS Glue を組み合わせて使用できます。

Microsoft SQL Server データウェアハウスのソースとしての権限

ソースとして Microsoft SQL Server データウェアハウスに必要な特権を以下に示します。

- VIEW DEFINITION
- VIEW DATABASE STATE
- SELECT ON SCHEMA :: *<schema_name>*

前の例では、*<source_schema>* プレースホルダーをソースの source_schema の名前に置き換えます。

スキーマを変換する、各データベースの付与を繰り返します。

さらに、次の権限を付与し、マスターデータベースで権限を実行します。

- VIEW SERVER STATE

SQL Server データウェアハウスをソースとして使用する場合の制限

Microsoft SQL Server 並列データウェアハウス (PDW) をソースとして使用することは、現在サポートされていません。

ソースとしての SQL Server データウェアハウスへの接続

以下の手順を使用して、SQL Server データウェアハウスソースデータベースを AWS Schema Conversion Tool に接続します。

SQL Server データウェアハウスソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Microsoft SQL Server]、次に [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - Microsoft SQL Server ソースデータウェアハウス接続情報を手動で入力するには、次の手順に従います。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームサービス (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
Instance name	SQL Server データウェアハウスのインスタンス名を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパス

パラメータ	アクション
	ワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー証明書を信頼する]: サーバー証明書を信頼するには、このオプションを選択します。 • [信頼ストア]: [グローバル設定] でセットアップした信頼ストア。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[SQL Server ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

SQL Server データウェアハウスから Amazon Redshift への変換設定

SQL Server データウェアハウスから Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、次に [変換設定] を選択します。上のリストから [Microsoft SQL Server] を選択し、

次に [Microsoft SQL Server – Amazon Redshift] を選択します。AWS SCT に、Microsoft SQL Server データウェアハウスから Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT での Microsoft SQL Server データウェアハウスから Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。
- ソーステーブルのパーティションを Amazon Redshift の別のテーブルに移行するには。そのためには、[UNION ALL ビューを使用する] を選択し、AWS SCT が 1 つのソーステーブルに対して作成できるターゲットテーブルの最大数を入力します。

  Amazon Redshift は、テーブルのパーティションをサポートしていません。この動作を工ミュレートしてクエリをより速く実行するために、AWS SCT は、ソーステーブルの各パーティションを

Amazon Redshift の個別のテーブルに移行できます。次に、AWS SCT はこれらすべてのテーブルのデータを含むビューを作成します。

AWS SCT は、ソーステーブルのパーティションの数を自動的に決定します。ソーステーブルパーティショニングのタイプによっては、この数は Amazon Redshift クラスターに適用できるテーブルのクォータを超える場合があります。このクォータに達しないようにするには、AWS SCT が 1 つのソーステーブルのパーティションに対して作成できるターゲットテーブルの最大数を入力します。デフォルトのオプションは 368 テーブルで、これは 1 年 366 日のパーティションと、NO RANGE および UNKNOWN パーティションの 2 つのテーブルを表します。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

SQL Server データウェアハウスから Amazon Redshift への変換最適化設定

SQL Server データウェアハウスから Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、次に [変換設定] を選択します。上のリストから [Microsoft SQL Server] を選択し、次に [Microsoft SQL Server — Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に SQL Server データウェアハウスから Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT の SQL Server データウェアハウスから Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。

- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。

  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

  選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

  [小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
  - [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
  - [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
  - [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

  また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT ソースとしての Teradata の使用

AWS SCT を使用して、スキーマ、コード・オブジェクトおよびアプリケーション・コードを Teradata から Amazon Redshift または Amazon Redshift に変換し、AWS Glue を組み合わせて使用できます。

ソースとしての Teradata の権限

ソースとして Teradata を使用するのに必要な権限を以下に示します。

- SELECT ON DBC
- SELECT ON SYSUDTLIB
- SELECT ON SYSLIB
- SELECT ON *<source_database>*
- CREATE PROCEDURE ON *<source_database>*

前述の例では、*<source_database>* プレースホルダーをソースデータベースの名前に置き換えます。

AWS SCT でソースデータベース内のすべてのプロシージャに対して HELP PROCEDURE を実行するには、CREATE PROCEDURE 権限が必要です。AWS SCT は、この権限を使用してソース Teradata データベースに新しいオブジェクトを作成することはありません。

ソースとしての Teradata への接続

以下の手順を使用して、Teradata ソースデータベースを AWS Schema Conversion Tool に接続します。

Teradata ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Teradata] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。

1. [AWS シークレット] で、シークレットの名前を選択します。
2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。

- Teradata ソースデータベースの接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション
接続名	データベースの名前を入力します。AWS SCT の左パネルのツリーにこの名前が表示されます。
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
データベース	Teradata データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。 AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。

パラメータ	アクション
[データの暗号化]	データベースと交換するデータを暗号化する場合は、このオプションを選択します。このオプションを選択すると、ポート番号 443 を使用して AWS SCT と Teradata データベースとの間で暗号化されたデータが転送されます。
[Teradata ドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Teradata ソースでの LDAP 認証の使用

Windows で Microsoft Active Directory を実行する Teradata ユーザーに対して Lightweight Directory Access Protocol (LDAP) 認証をセットアップするには、以下の手順を使用します。

以下の手順では、アクティブディレクトリドメインは `test.local.com` です。Windows サーバーは DC で、デフォルト設定で構成されています。以下のスクリプトは `test_ldap` Active Directory を作成し、このアカウントはパスワード `test_ldap` を使用します。

Microsoft Active Directory を Windows で実行する Teradata ユーザーに対して LDAP 認証を設定するには

1. `/opt/teradata/tdat/tdgss/site` ディレクトリで、ファイル `TdgssUserConfigFile.xml` を編集します。LDAP セクションを次のように変更します。

```
AuthorizationSupported="no"

LdapServerName="DC.test.local.com"
LdapServerPort="389"
```

```
LdapServerRealm="test.local.com"
LdapSystemFQDN="dc= test, dc= local, dc=com"
LdapBaseFQDN="dc=test, dc=local, dc=com"
```

2. 次のように設定を実行して変更を適用します。

```
#cd /opt/teradata/tdgss/bin
#./run_tdgssconfig
```

3. 次のコマンドを使用して設定をテストします。

```
# /opt/teradata/tdat/tdgss/14.10.03.01/bin/tdsbind -u test_ldap -w test_ldap
```

出力は以下のようになります。

```
LdapGroupBaseFQDN: dc=Test, dc=local, dc=com
LdapUserBaseFQDN: dc=Test, dc=local, dc=com
LdapSystemFQDN: dc= test, dc= local, dc=com
LdapServerName: DC.test.local.com
LdapServerPort: 389
LdapServerRealm: test.local.com
LdapClientUseTls: no
LdapClientTlsReqCert: never
LdapClientMechanism: SASL/DIGEST-MD5
LdapServiceBindRequired: no
LdapClientTlsCRLCheck: none
LdapAllowUnsafeServerConnect: yes
UseLdapConfig: no
AuthorizationSupported: no
FQDN: CN=test, CN=Users, DC=Anthem, DC=local, DC=com
AuthUser: ldap://DC.test.local.com:389/CN=test1,CN=Users,DC=test,DC=local,DC=com
DatabaseName: test
Service: tdsbind
```

4. 次のコマンドを使用して TPA を再起動します。

```
#tpareset -f "use updated TDGSSCONFIG GDO"
```

5. 次に示すように、Active Directory と同じユーザーを Teradata データベースに作成します。

```
CREATE USER test_ldap AS PERM=1000, PASSWORD=test_ldap;
GRANT LOGON ON ALL TO test WITH NULL PASSWORD;
```

LDAP ユーザー用に Active Directory でユーザーパスワードを変更する場合、LDAP モードで Teradata への接続中に、この新しいパスワードを指定します。DEFAULT モードでは、LDAP ユーザー名と任意のパスワードを使用して Teradata に接続します。

ソース Teradata データウェアハウスでの統計収集の設定

ソース Teradata データウェアハウスを変換するために、AWS SCT は統計を使用して変換された Amazon Redshift データウェアハウスを最適化します。AWS SCT で統計を収集することも、統計ファイルをアップロードすることもできます。詳細については、「統計の収集またはアップロード」を参照してください。

AWS SCT でデータウェアハウスから統計情報を収集できるようにするには、以下の前提条件となるタスクを完了してください。

Teradata データウェアハウスから統計を収集するには

1. 次のクエリを実行して、データウェアハウス内のすべてのテーブルの統計情報を収集します。

   ```
   collect summary statistics on table_name;
   ```

   上記の例では、*table_name* をソース テーブルの名前に置き換えます。変換するテーブルごとにこのクエリを繰り返します。

2. 次のクエリを実行して、データウェアハウスの変換に使用するユーザーのアカウント文字列を決定します。

   ```
   select * from dbc.accountinfo where username ='user_name'
   ```

3. 前の例のアカウント文字列を使用して、特定のユーザーのクエリロギングを有効にします。

   ```
   BEGIN QUERY LOGGING WITH OBJECTS, SQL ON ALL ACCOUNT=('$M$BUSI$S$D$H');
   ```

   または、すべてのデータベースユーザーのクエリログを有効にします。

   ```
   BEGIN QUERY LOGGING WITH SQL, OBJECTS LIMIT SQLTEXT=0 ON ALL;
   ```

データウェアハウス統計の収集が完了したら、クエリログをオフにします。以下のコード例を使用してこれを行うことができます。

```
end query logging with explain, objects, sql on all account=(' $M$BUSI$S$D$H');
```

ソース Teradata データウェアハウスからオフラインモードで統計を収集します。

Teradata データウェアハウスで統計収集を設定したら、AWS SCT プロジェクトで統計を収集できます。また、ベーシック Teradata クエリ (BTEQ) スクリプトを使用してオフラインモードで統計情報を収集することもできます。その後、収集した統計を含むファイルを AWS SCT プロジェクトにアップロードできます。詳細については、「統計の収集またはアップロード」を参照してください。

Teradata データウェアハウスからオフラインモードで統計情報を収集するには

1. 次の内容で、off-line_stats.bteq スクリプトを作成します。

```
.OS IF EXIST column-stats-tera.csv del /F column-stats-tera.csv
.OS IF EXIST table-stats-tera.csv del /F table-stats-tera.csv
.OS IF EXIST column-skew-script-tera.csv del /F column-skew-script-tera.csv
.OS IF EXIST column-skew-stats-tera.csv del /F column-skew-stats-tera.csv
.OS IF EXIST query-stats-tera.csv  del /F query-stats-tera.csv
.LOGON your_teradata_server/your_login, your_password
.EXPORT REPORT FILE = table-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

SELECT
    '"' || OREPLACE(COALESCE(c.DatabaseName, ''), '"', '""') || '";' ||
    '"' || OREPLACE(COALESCE(c.TableName, ''), '"', '""') || '";' ||
    '"' || TRIM(COALESCE(s.reference_count, '0')) || '";' ||
    '"' || TRIM(COALESCE(CAST(p.RowCount AS BIGINT), '0')) || '";' ||
    '"' || CAST(CAST(w.size_in_mb AS DECIMAL (38,1) FORMAT 'Z9.9') AS VARCHAR(38))
 || '";' ||
    '"' || TRIM(COALESCE(r.stat_fk_dep_count, '0')) || '";' ||
    '"' || CAST(CAST(current_timestamp(0) as timestamp(0) format 'YYYY-MM-
DDBHH:MI:SS') as VARCHAR(19)) || '"'
(TITLE
 '"database_name";"table_name";"reference_count";"row_count";"size_in_mb";"stat_fk_dep_coun
FROM (select databasename, tablename
        from DBC.tablesv
        where tablekind IN ('T','O')
        and databasename = 'your_database_name'
         ) c
left join
        (select DatabaseName, TableName, max(RowCount) RowCount
```

```
        from dbc.tableStatsv
        group by 1,2)p
on p.databasename = c.databasename
and p.tablename = c.tablename
left join
        (SELECT r.ChildDB as DatabaseName,
        r.ChildTable as TableName,
        COUNT(DISTINCT r.ParentTable) reference_count
        FROM DBC.All_RI_ChildrenV r
        GROUP BY r.ChildDB, r.ChildTable) s
on s.databasename = c.databasename
and s.tablename = c.tablename
left join
        (SELECT r.ParentDB as DatabaseName,
        r.ParentTable as TableName,
        COUNT(DISTINCT r.ChildTable) stat_fk_dep_count
        FROM DBC.All_RI_ParentsV r
        GROUP BY r.ParentDB, r.ParentTable) r
on r.databasename = c.databasename
and r.tablename = c.tablename
left join
        (select databasename, tablename,
        sum(currentperm)/1024/1024 as size_in_mb
        from dbc.TableSizeV
        group by 1,2) w
on w.databasename = c.databasename
and w.tablename = c.tablename
WHERE COALESCE(r.stat_fk_dep_count,0) + COALESCE(CAST(p.RowCount AS BIGINT),0) +
 COALESCE(s.reference_count,0) > 0;

.EXPORT RESET

.EXPORT REPORT FILE = column-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000
    '"' || TRIM(COALESCE(CAST(t2.card AS BIGINT), '0')) || '";' ||

SELECT
 '"' || OREPLACE(COALESCE(trim(tv.DatabaseName), ''), '"', '""') || '";' ||
     '"' || OREPLACE(COALESCE(trim(tv.TableName), ''), '"', '""') || '";' ||
 '"' || OREPLACE(COALESCE(trim(tv.columnname), ''), '"', '""') || '";' ||
                         '"' || TRIM(COALESCE(CAST(t2.card AS BIGINT), '0')) ||
 '";' ||
```

```
 '"' || CAST(current_timestamp AS VARCHAR(19)) || '"' (TITLE
 '"database_name";"table_name";"column_name";"cardinality";"current_ts"')
FROM dbc.columnsv tv
LEFT JOIN
(
 SELECT
  c.DatabaseName AS DATABASE_NAME,
  c.TABLENAME  AS TABLE_NAME,
  c.ColumnName AS COLUMN_NAME,
  c.UniqueValueCount AS CARD
 FROM dbc.tablestatsv c
 WHERE c.DatabaseName = 'your_database_name'
 AND c.RowCount <> 0
) t2
ON tv.DATABASENAME = t2.DATABASE_NAME
AND tv.TABLENAME = t2.TABLE_NAME
AND tv.COLUMNNAME = t2.COLUMN_NAME
WHERE t2.card > 0;

.EXPORT RESET

.EXPORT REPORT FILE = column-skew-script-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

SELECT
'SELECT CAST(''"' || TRIM(c.DatabaseName) || '";"' || TRIM(c.TABLENAME)  || '";"'
 || TRIM(c.COLUMNNAME) || '";"'' ||
TRIM(CAST(COALESCE(MAX(cnt) * 1.0 / SUM(cnt), 0) AS NUMBER FORMAT ''9.9999'')) ||
 '"";"'' ||
CAST(CURRENT_TIMESTAMP(0) AS VARCHAR(19)) || '"'' AS VARCHAR(512))
AS """DATABASE_NAME"";""TABLE_NAME"";""COLUMN_NAME"";""SKEWED"";""CURRENT_TS"""
FROM(
SELECT COUNT(*) AS cnt
FROM "' || c.DATABASENAME || '"."' || c.TABLENAME ||
'" GROUP BY "' || c.COLUMNNAME || '") t' ||
 CASE WHEN ROW_NUMBER() OVER(PARTITION BY c.DATABASENAME
 ORDER BY c.TABLENAME DESC, c.COLUMNNAME DESC) <> 1
 THEN ' UNION ALL'
 ELSE ';' END (TITLE '--SKEWED--')
FROM dbc.columnsv c
INNER JOIN
(SELECT databasename, TABLENAME
FROM dbc.tablesv  WHERE tablekind = 'T'
```

```
AND  databasename = 'your_database_name') t
ON t.databasename = c.databasename
AND t.TABLENAME = c.TABLENAME
INNER JOIN
(SELECT databasename, TABLENAME, columnname FROM  dbc.indices GROUP BY 1,2,3
WHERE  TRANSLATE_CHK (databasename USING LATIN_TO_UNICODE) + TRANSLATE_CHK
 (TABLENAME USING LATIN_TO_UNICODE) + TRANSLATE_CHK (columnname USING
 LATIN_TO_UNICODE) = 0
) i
ON i.databasename = c.databasename
AND i.TABLENAME = c.TABLENAME
AND i.columnname = c.columnname
WHERE c.ColumnType NOT IN ('CO','JN','N','++','VA','UT','AN','XM','A1','BO')
ORDER BY c.TABLENAME, c.COLUMNNAME;

.EXPORT RESET

.EXPORT REPORT FILE = column-skew-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

.RUN FILE = column-skew-script-tera.csv

.EXPORT RESET

.EXPORT REPORT FILE = query-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 32000

SELECT
  '"' || RTRIM(CAST(SqlTextInfo AS VARCHAR(31900)), ';') || '";"' ||
  TRIM(QueryCount) || '";"' ||
  TRIM(QueryId) || '";"' ||
  TRIM(SqlRowNo) || '";"' ||
  TRIM(QueryParts) || '";"' ||
  CAST(CURRENT_TIMESTAMP(0) AS VARCHAR(19)) || '"'
(TITLE
 '"query_text";"query_count";"query_id";"sql_row_no";"query_parts";"current_ts"')
  FROM
  (
    SELECT  QueryId,  SqlTextInfo, SqlRowNo, QueryParts, QueryCount,
    SUM(QueryFirstRow) OVER (ORDER BY QueryCount DESC, QueryId ASC, SqlRowNo ASC
    ROWS UNBOUNDED PRECEDING) AS topN
    FROM
```

```
    (SELECT QueryId,  SqlTextInfo, SqlRowNo, QueryParts, QueryCount,
      CASE WHEN
      ROW_NUMBER() OVER (PARTITION BY QueryCount, SqlTextInfo ORDER BY QueryId,
 SqlRowNo) = 1 AND SqlRowNo = 1
    THEN 1 ELSE 0 END AS QueryFirstRow
    FROM (
      SELECT q.QueryId,  q.SqlTextInfo, q.SqlRowNo,
      MAX(q.SqlRowNo) OVER (PARTITION BY q.QueryId) QueryParts,
      COUNT(q.SqlTextInfo) OVER (PARTITION BY q.SqlTextInfo) QueryCount
      FROM DBC.dbqlsqltbl q
      INNER JOIN
      (
        SELECT QueryId
        FROM DBC.DBQLogTbl t
        WHERE TRIM(t.StatementType) IN ('SELECT')
        AND TRIM(t.AbortFlag) = '' AND t.ERRORCODE = 0
        AND  (CASE WHEN 'All users' IN ('All users') THEN 'All users' ELSE
 TRIM(t.USERNAME) END) IN ('All users') --user_name list
        AND t.StartTime > CURRENT_TIMESTAMP - INTERVAL '30' DAY
        GROUP BY 1
      ) t
      ON q.QueryId = t.QueryId
      INNER JOIN
      (
        SELECT QueryId
        FROM DBC.QryLogObjectsV
        WHERE ObjectDatabaseName = 'your_database_name'
        AND ObjectType = 'Tab'
        AND CollectTimeStamp > CURRENT_TIMESTAMP - INTERVAL '30' DAY
        GROUP BY 1
      ) r
      ON r.QueryId = t.QueryId
      WHERE q.CollectTimeStamp > CURRENT_TIMESTAMP - INTERVAL '30' DAY
    ) t
  ) t
  WHERE SqlTextInfo NOT LIKE '%";"%'
  ) q
  WHERE
  QueryParts >=1
  AND topN <= 50
  ORDER BY QueryCount DESC, QueryId, SqlRowNo
  QUALIFY COUNT(QueryId) OVER (PARTITION BY QueryId) = QueryParts;

.EXPORT RESET
```

```
.LOGOFF

.QUIT
```

2. 前のステップで作成した BTEQ スクリプトを実行する td_run_bteq.bat ファイルを作成します。このファイルには以下の内容を使用します。

```
@echo off > off-line_stats1.bteq & setLocal enableDELAYedexpansion
@echo off > off-line_stats2.bteq & setLocal enableDELAYedexpansion

set old1=your_teradata_server
set new1=%1
set old2=your_login
set new2=%2
set old3=your_database_name
set new3=%3
set old4=your_password
set /p new4=Input %2 pass?

for /f "tokens=* delims= " %%a in (off-line_stats.bteq) do (
set str1=%%a
set str1=!str1:%old1%=%new1%!
>> off-line_stats1.bteq echo !str1!
)

for /f "tokens=* delims= " %%a in (off-line_stats1.bteq) do (
set str2=%%a
set str2=!str2:%old2%=%new2%!
>> off-line_stats2.bteq echo !str2!
)

type nul > off-line_stats1.bteq

for /f "tokens=* delims= " %%a in (off-line_stats2.bteq) do (
set str3=%%a
set str3=!str3:%old3%=%new3%!
>> off-line_stats1.bteq echo !str3!
)

type nul > off-line_stats2.bteq

for /f "tokens=* delims= " %%a in (off-line_stats1.bteq) do (
```

```
set str4=%%a
set str4=!str4:%old4%=%new4%!
>> off-line_stats2.bteq echo !str4!
)

del .\off-line_stats1.bteq

echo export starting...

bteq -c UTF8 < off-line_stats.bteq > metadata_export.log

pause
```

3. 前のステップで作成したバッチファイルを実行する runme.bat ファイルを作成します。このファイルには以下の内容を使用します。

   ```
   .\td_run_bteq.bat ServerName UserName DatabaseName
   ```

   runme.bat ファイルの [*ServerName*]、[*UserName*]、[*DatabaseName*] を適切な値に置き換えます。

   次に、runme.bat ファイルを実行します。Amazon Redshift に変換するデータウェアハウスごとに、このステップを繰り返します。

このスクリプトを実行すると、各データベースの統計情報を含む 3 つのファイルが返されます。これらのファイルは AWS SCT プロジェクトにアップロードできます。このためには、プロジェクトの左側のパネルからデータウェアハウスを選択し、右クリックによりコンテキストメニューを開きます。[統計をアップロード] を選択します。

Teradata から Amazon Redshift への変換設定

Teradata から Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Teradata] を選択し、次に [Teradata – Amazon Redshift] を選択します。AWS SCT に、Teradata から Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT での Teradata から Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

[変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。

- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブルの数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。

- ソーステーブルのパーティションを Amazon Redshift の別のテーブルに移行するには。そのためには、[UNION ALL ビューを使用する] を選択し、AWS SCT が 1 つのソーステーブルに対して作成できるターゲットテーブルの最大数を入力します。

  Amazon Redshift は、テーブルのパーティションをサポートしていません。この動作をエミュレートしてクエリをより速く実行するために、AWS SCT は、ソーステーブルの各パーティションを Amazon Redshift の個別のテーブルに移行できます。次に、AWS SCT はこれらすべてのテーブルのデータを含むビューを作成します。

  AWS SCT は、ソーステーブルのパーティションの数を自動的に決定します。ソーステーブルパーティショニングのタイプによっては、この数は Amazon Redshift クラスターに適用できるテーブルのクォータを超える場合があります。このクォータに達しないようにするには、AWS SCT が 1 つのソーステーブルのパーティションに対して作成できるターゲットテーブルの最大数を入力し

ます。デフォルトのオプションは 368 テーブルで、これは 1 年 366 日のパーティションと、NO RANGE および UNKNOWN パーティションの 2 つのテーブルを表します。

- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

- 変換後のコードで明示的な列リストを SELECT * ステートメントに使用するには、[明示的な列宣言を使用する] を選択します。

- Amazon Redshift クラスターのプライマリキーとユニークキーの動作をエミュレートするには、[プライマリキーとユニークキーの動作をエミュレートする] を選択します。

  Amazon Redshift はユニークキーやプライマリキーを強制せず、情報提供のみを目的として使用します。これらの制約をコードで使用する場合は、AWS SCT が変換後のコードでもその動作をエミュレートすることを確認してください。

- ターゲット Amazon Redshift テーブル内のデータの一意性を確保するには。そのためには、[SET テーブルの動作をエミュレート] を選択します。

  Teradata は、SET 構文要素をデフォルトオプションとして使用してテーブルを作成します。SET テーブルに重複する行を追加することはできません。ソースコードでこの一意性制約が使用されていない場合は、このオプションをオフにしてください。この場合、変換されたコードの処理が速くなります。

  ソースコードで一意性制約としてテーブル内の SET オプションを使用している場合は、このオプションをオンにしてください。この場合、変換されたコードの INSERT..SELECT ステートメントは、AWS SCT によってソースデータベースの動作をエミュレートするように書き直されます。

Terata から Amazon Redshift への変換最適化設定

Teradata から Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Teradata] を選択し、次に [Teradata – Amazon

Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に Teradata から Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT での Teradata から Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。
- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。

  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

  選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

  [小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。
- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。

  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。

- [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
- [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
- [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT ソースとしての Vertica の使用

AWS SCT を使用して、Vertica からのスキーマ、コードオブジェクトおよびアプリケーションコードを Amazon Redshift に変換できます。

ソースとしての Vertica の権限

ソースとして Vertica を使用するのに必要な権限を以下に示します。

- USAGE ON SCHEMA *<schema_name>*
- USAGE ON SCHEMA PUBLIC
- SELECT ON ALL TABLES IN SCHEMA *<schema_name>*
- SELECT ON ALL SEQUENCES IN SCHEMA *<schema_name>*
- EXECUTE ON ALL FUNCTIONS IN SCHEMA *<schema_name>*
- EXECUTE ON PROCEDURE *<schema_name.procedure_name(procedure_signature)>*

上記の例で、次のプレースホルダを置き換えます。

- *schema_name* をソーススキーマの名前に置き換えます。
- *procedure_name* をソースプロシージャの名前に置き換えます。変換するプロシージャごとに、同じ権限を繰り返し付与します。
- *procedure_signature* を、プロシージャの引数タイプをカンマで区切ったリストに置き換えます。

ソースとしての Vertica への接続

以下の手順を使用して、Vertica ソースデータベースを AWS Schema Conversion Tool に接続します。

Vertica ソースデータベースに接続するには

1. AWS Schema Conversion Tool で、[Add source] (ソースの追加) を選択します。
2. [Vertica] を選択し、その後 [Next] (次へ) を選択します。

   [Add source] (ソースの追加) ダイアログボックスが表示されます。
3. [接続名] にデータベースの名前を入力します。この名前が AWS SCT の左側のパネルのツリーに表示されます。
4. AWS Secrets Manager からのデータベース認証情報を使用するか、手動で入力してください。

   - Secrets Manager のデータベース認証情報を使用するには、以下の手順に従ってください。
     1. [AWS シークレット] で、シークレットの名前を選択します。
     2. [入力] を選択すると、Secrets Manager のデータベース接続ダイアログボックスにすべての値が自動的に入力されます。

     Secrets Manager のデータベース認証情報を使用する方法については、「AWS Secrets Manager を使用する」を参照してください。
   - Vertica ソースデータベースの接続情報を手動で入力するには、以下の手順に従ってください。

パラメータ	アクション
[Server name] (サーバー名)	ソースデータベースサーバーのドメインネームシステム (DNS) 名または IP アドレスを入力します。
Server port	ソースデータベースサーバーへの接続に使用するポートを入力します。
データベース	Vertica データベースの名前を入力します。
User name (ユーザー名) と [Password] (パスワード)	データベース認証情報を入力して、ソースデータベースサーバーに接続します。

パラメータ	アクション
	AWS SCT でパスワードを使用して、プロジェクト内のデータベースに接続することを選択する場合にのみソースデータベースに接続します。ソースデータベースのパスワードの漏洩を防ぐため、デフォルトで AWS SCT にパスワードは保存されません。AWS SCT プロジェクトを閉じて再び開いた場合は、必要に応じて、ソースデータベースへの接続に使用するパスワードの入力を求められます。
SSL の使用	データベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、このオプションを選択します。[SSL] タブで、必要に応じて、以下の追加情報を提供します。 • [サーバー証明書を確認する]: 信頼ストアを使用してサーバー証明書を確認するには、このオプションを選択します。 • [信頼ストア]: [グローバル設定] でセットアップした信頼ストア。 • [キーストア]: [グローバル設定] でセットアップしたキーストア。
Store Password	AWS SCT は、安全なボールトを作成して、SSL 証明書とデータベースパスワードを保存します。このオプションをオンにすると、データベースのパスワードが保存されるため、パスワードを入力しなくてもデータベースにすばやく接続できます。
[Vertica のドライバーパス]	ソースデータベースへの接続に使用するドライバのパスを入力します。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。 ドライバパスをグローバルプロジェクト設定に保存する場合、ドライバパスは接続ダイアログボックスに表示されません。詳細については、「グローバル設定でのドライバパスの保存」を参照してください。

5. [Test Connection] (接続のテスト) を選択して、AWS SCT がソースデータベースに正常に接続できることを確認します。
6. [Connect] (接続) を選択して、ソースデータベースに接続します。

Vertica から Amazon Redshift への変換設定

Vertica から Amazon Redshift への変換設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Vertica] を選択し、次に [Vertica – Amazon Redshift] を選択します。AWS SCT に、Vertica から Amazon Redshift への変換に使用可能なすべての設定が表示されます。

AWS SCT での Vertica から Amazon Redshift への変換設定には、以下のオプションが含まれています。

- 変換されたコード内のアクション項目に関するコメントの数を制限する。

  [変換後のコードにコメントを追加] で、選択した重要度以上のアクションアイテムについて、アクションアイテムの重要度を選択します。AWS SCT は、選択した重要度以上のアクションアイテムについて、変換後のコードにコメントを追加します。

  例えば、変換済みのコード内のコメント数を最小限に抑えるには、[エラーのみ] を選択します。変換済みのコードにすべてのアクション項目のコメントを含めるには、[すべてのメッセージ] を選択します。
- AWS SCT でターゲット Amazon Redshift クラスターに適用できるテーブルの最大数を設定します。

  [ターゲット Amazon Redshift クラスターの最大テーブル数] では、AWS SCT が Amazon Redshift クラスターに適用できるテーブルの数を選択します。

  Amazon Redshift には、クラスターノードタイプの使用を制限するクォータがあります。[自動] を選択した場合、AWS SCT はノードタイプに応じてターゲット Amazon Redshift クラスターに適用するテーブルの数を決定します。オプションで、値を手動で選択します。詳細については、「Amazon Redshift 管理ガイド」の「Amazon Redshift のクォータと制限」を参照してください。

  Amazon Redshift クラスターが保存できる量を超える場合でも、AWS SCT はすべてのソーステーブルを変換します。AWS SCT では、変換されたコードはプロジェクトに保存され、ターゲットデータベースには適用されません。変換したコードを適用したときにテーブルの Amazon Redshift クラスターのクォータに達すると、AWS SCT に警告メッセージが表示されます。また、テーブル

の数が制限に達するまで、AWS SCT はターゲット Amazon Redshift クラスターにテーブルを適用します。

- ソーステーブルのパーティションを Amazon Redshift の別のテーブルに移行するには。そのためには、[UNION ALL ビューを使用する] を選択し、AWS SCT が 1 つのソーステーブルに対して作成できるターゲットテーブルの最大数を入力します。

  Amazon Redshift は、テーブルのパーティションをサポートしていません。この動作をエミュレートしてクエリをより速く実行するために、AWS SCT は、ソーステーブルの各パーティションを Amazon Redshift の個別のテーブルに移行できます。次に、AWS SCT はこれらすべてのテーブルのデータを含むビューを作成します。

  AWS SCT は、ソーステーブルのパーティションの数を自動的に決定します。ソーステーブルパーティショニングのタイプによっては、この数は Amazon Redshift クラスターに適用できるテーブルのクォータを超える場合があります。このクォータに達しないようにするには、AWS SCT が 1 つのソーステーブルのパーティションに対して作成できるターゲットテーブルの最大数を入力します。デフォルトのオプションは 368 テーブルで、これは 1 年 366 日のパーティションと、NO RANGE および UNKNOWN パーティションの 2 つのテーブルを表します。
- Amazon Redshift テーブルの列に圧縮を適用します。そのためには、[圧縮エンコードを使用] を選択します。

  AWS SCT は、デフォルトの Amazon Redshift アルゴリズムを使用して、圧縮エンコーディングを列に自動的に割り当てます。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。

  デフォルトでは、Amazon Redshift はソートキーと分散キーとして定義されている列に圧縮を適用しません。この動作を変更したり、これらの列に圧縮を適用したりできます。そのためには、[KEY 列には圧縮エンコードを使用] を選択します。このオプションは、[圧縮エンコードを使用] オプションを選択した場合にのみ選択できます。

Vertica から Amazon Redshift への変換最適化設定

Vertica から Amazon Redshift への変換最適化設定を編集するには、AWS SCT で [設定] を選択し、[変換設定] を選択します。上のリストから [Vertica] を選択し、次に [Vertica – Amazon Redshift] を選択します。左側のペインで、[最適化戦略] を選択します。AWS SCT に Vertica から Amazon Redshift への変換の変換最適化設定が表示されます。

AWS SCT での Vertica から Amazon Redshift への変換最適化設定には、以下のオプションが含まれています。

- 自動テーブル最適化を使用するには。そのためには、[Amazon Redshift の自動テーブルチューニングを使用する] を選択します。

  自動テーブル最適化は、テーブルの設計を自動的に最適化する Amazon Redshift の自己調整プロセスです。詳細については、『Amazon Redshift データベース開発者ガイド』の「自動テーブル最適化の操作」を参照してください。

  自動テーブル最適化のみを利用するには、[初期キー選択戦略] で [なし] を選択します。
- 戦略を使用してソートキーと分散キーを選択するには。

  Amazon Redshift メタデータ、統計情報、またはこれらのオプションの両方を使用して、ソートキーと配布キーを選択できます。[最適化戦略] タブの [初期キー選択戦略] では、以下のいずれかのオプションを選択します。
  - メタデータを使用し、統計情報は無視する
  - メタデータを無視し、統計情報を使用する
  - メタデータと統計情報を使用する

  選択したオプションに応じて、最適化戦略を選択できます。次に、各戦略について、値 (0 ~ 100) を入力します。これらの値は各戦略の重みを定義します。これらの重み値を使用して、AWS SCT は各ルールがディストリビューションキーとソートキーの選択にどのように影響するかを定義します。デフォルト値は、AWS 移行のベストプラクティスに基づいています。

  [小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力して小さいテーブルとして定義します。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。
- 戦略の詳細を設定するには。

  各最適化戦略の重みを定義することに加えて、最適化設定を構成できます。そのためには、[変換の最適化] を選択します。
  - [ソートキー列の制限] には、ソートキーの列の最大数を入力します。
  - [歪んだしきい値] には、列の歪んだ値のパーセンテージ (0 ~ 100) を入力します。AWS SCT は、歪み値がしきい値より大きい列を分散キーの候補リストから除外します。AWS SCT では、列の歪んだ値をレコードの総数に対する最も一般的な値の出現回数の割合として定義します。
  - [クエリ履歴テーブルの上位 N 件のクエリ] には、分析対象として最も頻繁に使用されるクエリの数 (1 ~ 100) を入力します。
  - [統計ユーザーを選択] では、クエリ統計を分析したいデータベースユーザーを選択します。

また、[最適化戦略」タブでは、[小さいテーブルを検索] 戦略の小さいテーブルのサイズを定義できます。[最小テーブル行数] と [最大テーブル行数] には、テーブル内の最小行数と最大行数を入力すると、小さいテーブルとして考慮されます。AWS SCT は ALL 分散スタイルを小さなテーブルに適用します。この場合は、テーブル全体のコピーがすべてのノードに分散されます。

AWS SCT でのマッピングルールの作成

複数のソースとターゲットデータベースサーバーを単一の AWS SCT プロジェクトに追加することができます。これにより、複数のデータベースを異なるターゲットプラットフォームに移行するときに、プロジェクトの管理が簡素化されます。

新しいプロジェクトを作成し、ソースとターゲットデータベースを追加したら、マッピングルールを作成します。AWS SCT では、移行評価レポートを作成し、データベーススキーマを変換するには、少なくとも 1 つのマッピングルールが必要です。

マッピングルールにはペアが記述されており、これにはソースデータベーススキーマまたはソースデータベースとターゲットデータベースプラットフォームが含まれます。単一の AWS SCT プロジェクト内に、複数のマッピングルールを作成できます。マッピングルールを使用して、すべてのソースデータベーススキーマを適切なターゲットデータベースプラットフォームに変換します。

変換されたコードのスキーマの名前を変更するには、移行ルールを設定します。たとえば、移行ルールでは、スキーマの名前を変更したり、オブジェクト名にプレフィックスを追加したり、列の照合順序を変更したり、データ型を変更したりできます。変換したコードにこれらの変更を適用するには、ソーススキーマを変換する前に必ず移行ルールを作成してください。詳細については、「移行ルールの作成」を参照してください。

マッピングルールを作成できるのは、サポートされているデータベース変換ペアに対してのみです。サポートされている変換ペアのリストについては、「AWS SCT のソース」を参照してください。

AWS SCT バージョン 1.0.655 以前で保存したプロジェクトを開くと、AWS SCT は、すべてのソースデータベーススキーマのターゲットデータベースプラットフォームに対するマッピングルールを自動的に作成します 他のターゲットデータベースプラットフォームを追加するには、既存のマッピングルールを削除してから、新しいマッピングルールを作成します。

トピック

- 新しいマッピングルールの追加
- マッピングルールの管理
- 仮想ターゲットの使用
- 単一の AWS SCT プロジェクトで複数のサーバーを使用する場合の制限

新しいマッピングルールの追加

単一のプロジェクト内に複数のマッピングルールを作成できます。AWS SCT には、マッピングルールがプロジェクトの一部として保存されます。プロジェクトを開いた状態で、以下の手順で新しいマッピングルールを作成します。

マッピングルールを作成するには

1. [View] (ビュー) メニューで、[Mapping View] (マッピングビュー) を選択します。
2. 左側のパネルで、マッピングルールに追加するスキーマまたはデータベースを選択します。
3. 右側のパネルで、選択したソーススキーマまたはデータベースのターゲットデータベースプラットフォームを選択します。

   仮想データベースプラットフォームをターゲットとして選択できます。詳細については、「仮想ターゲットの使用」を参照してください。
4. [Create mapping] (マッピングの作成) を選択します。

   AWS SCT の [Server mappings] (サーバーマッピング) リストに、この新しいマッピングルールが追加されます。

すべての変換ペアに対して、マッピングルールを追加します。評価レポートを作成するか、データベーススキーマを変換するには、[View] (ビュー) メニューの [Main view] (メインビュー) を選択します。

マッピングルールに含まれるすべてのスキーマオブジェクトが、AWS SCT で強調表示されます。

マッピングルールの管理

既存のマッピングルールを削除するか、新しいマッピングルールを AWS Schema Conversion Tool (AWS SCT) プロジェクトに追加します。

ソースデータベース全体のマッピングルールを作成すると、AWS SCT でソースデータベーススキーマごとに 1 つのマッピングルールが作成されます。数十のスキーマやデータベースを含むプロジェクトでは、特定のスキーマにどのターゲットが使用されているのか理解するのが難しい場合があります。スキーマのマッピングルールをすばやく見つけるには、AWS SCT で以下のフィルタオプションを 1 つまたは複数使用してください。

マッピングルールをフィルタリングするには

1. [ビュー] メニューで、[マッピングビュー] を選択します。
2. [ソースデータベース] で、ソースデータベースを選択します。

   フィルターのデフォルトは [すべて] です。つまり、AWS SCT にすべてのソースデータベースのマッピングルールが表示されます。
3. [ソーススキーマ] には、ソーススキーマ名を入力します。パーセント (%) をワイルドカードとして使用して、スキーマ名に含まれる任意の記号をいくつでも置き換えます。

   フィルタのデフォルトは [%] ワイルドカードです。つまり、AWS SCT に、すべてのソースデータベーススキーマ名のマッピングルールが表示されます。
4. [移行ルールあり] で [はい] を選択すると、データ移行ルールが作成されたマッピングルールが表示されます。データ移行ルールを含まないマッピングルールを表示するには、[いいえ] を選択します。詳細については、「でのデータ移行ルールの作成 AWS SCT」を参照してください。

   フィルタのデフォルトは [すべて] で、AWS SCT にすべてのマッピングルールが表示されます。
5. [ターゲットサーバー] で、ターゲットデータベースを選択します。

   フィルタのデフォルトは [すべて] で、AWS SCT にすべてのターゲットデータベースのマッピングルールが表示されます。

プロジェクトを開いた状態で、以下の手順でマッピングルールを削除します。マッピングルールを追加する詳細については、「新しいマッピングルールの追加」を参照してください。

マッピングルールを削除するには

1. [View] (ビュー) メニューで、[Mapping View] (マッピングビュー) を選択します。
2. [Server mappings] (サーバーマッピング) の場合、削除するマッピングルールを選択します。
3. [Delete selected mappings] (選択したマッピングを削除) を選択します。

   AWS SCT で選択したマッピングルールが削除されます。

仮想ターゲットの使用

AWS SCT でソースデータベーススキーマが、サポートされている任意のターゲットデータベースプラットフォームにどのように変換されるのかを確認できます。これを行うにあたり、既存のター

ゲットデータベースに接続する必要はありません。代わりに、マッピングルールを作成するときに、右側のパネルで仮想ターゲットデータベースプラットフォームを選択できます。詳細については、「新しいマッピングルールの追加」を参照してください。右側のパネルで [サーバー]、[NoSQL クラスター]、および [ETL] ノードを展開して、仮想ターゲットデータベースプラットフォームのリストを確認してください。

AWS SCT は、次の仮想ターゲットデータベースプラットフォームをサポートしています。

- Amazon Aurora MySQL 互換エディション
- Amazon Aurora PostgreSQL 互換エディション
- Amazon DynamoDB
- Amazon Redshift
- Amazon Redshift および AWS Glue
- AWS Glue
- AWS Glue Studio
- Babelfish for Aurora PostgreSQL
- MariaDB
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL

Babelfish for Aurora PostgreSQL をターゲットデータベースプラットフォームとして使用する場合、作成できるのはデータベース移行評価レポートのみです。詳細については、「the section called “移行評価レポート”」を参照してください。

仮想ターゲットデータベースプラットフォームを使用する場合、変換されたコードをファイルに保存できます。詳細については、「the section called “変換されたスキーマの保存”」を参照してください。

単一の AWS SCT プロジェクトで複数のサーバーを使用する場合の制限

単一の AWS SCT プロジェクト内の複数のサーバーを使用してスキーマを変換する場合は、以下の制限が適用されます。

- プロジェクトに同じサーバーを追加できるのは、1 回のみです。
- サーバースキーマを特定のターゲットスキーマにマップすることはできません。ターゲットサーバーにのみマップできます。AWS SCT が変換中にターゲットスキーマを作成します。
- 下位レベルのソースオブジェクトをターゲットサーバーにマップすることはできません。
- 1 つのソーススキーマをプロジェクト内の 1 つのターゲットサーバーにのみマッピングできます。
- 評価レポートの作成、スキーマの変換、またはデータの抽出を行うには、必ずソースをターゲットサーバーにマッピングしてください。

変換レポートの作成

データベース変換を計画しているときは、関連する内容を理解するのに役立ついくつかのレポートを作成すると便利です。AWS Schema Conversion Tool または を使ってレポートを作成できます。

AWS SCT を使用してデータベース移行評価レポートを作成できます。このレポートを使用すると、スキーマ変換タスクの概要やターゲットデータベースに自動的に変換できない項目の詳細を取得できます。この報告書を利用すれば、AWS SCT を使用してプロジェクトをどのくらい完了できるのか、また変換を完了するためには他にどんな必要となる事項があるのかを評価できます。評価レポートを作成するには、AWS SCT のデータベースのコンテキスト (右クリック) メニューから [Create Report] (レポートの作成) を使用します。

トピック

- AWS SCT を使用した移行評価レポートの作成

AWS SCT を使用した移行評価レポートの作成

AWS Schema Conversion Tool の重要な点は、スキーマ変換の複雑さを見積もるために生成される評価レポートです。このデータベース移行評価レポートは、ターゲット DB インスタンスの DB エンジンに変換できないスキーマに関する、すべてのスキーマ変換タスクやアクション項目の詳細をまとめたものです。レポートはアプリケーションで表示することも、カンマ区切り値 (CSV) ファイルまたは PDF ファイルとしてエクスポートすることもできます。

1 つのプロジェクトに複数のソースデータベースとターゲットデータベースを追加すると、AWS SCT で、すべてのコンバージョンペアのレポートが 1 つのデータベース移行評価レポートに集約されます。

仮想ターゲットデータベースプラットフォームを使用して評価レポートを生成し、選択したデータベースプラットフォームへの移行の複雑さを把握できます。この場合、ターゲットデータベースプラットフォームに接続する必要はありません。たとえば、Babelfish for Aurora PostgreSQL を仮想ターゲットデータベースプラットフォームとして使用して、データベース移行評価レポートを作成できます。仮想ターゲットデータベースプラットフォームの詳細については、「the section called “仮想ターゲット”」を参照してください。

移行評価レポートには、次のものが含まれます。

- エグゼクティブサマリー

- ライセンス評価
- クラウドサポート。ソースデータベースの機能のうちターゲットで使用できないものを示します。
- 推奨事項。サーバーオブジェクトの変換、バックアップの提案、リンクされたサーバーの変更などです。

このレポートには、自動的に変換できないため、ターゲット DB インスタンスで同等のコードを記述するために必要な労力の予測も含まれます。

AWS SCT を使用して、Amazon RDS DB インスタンスに既存のスキーマを移行する場合は、レポートを使用して、AWS クラウドへの移動やライセンスタイプの変更に関する要件を分析できます。

トピック

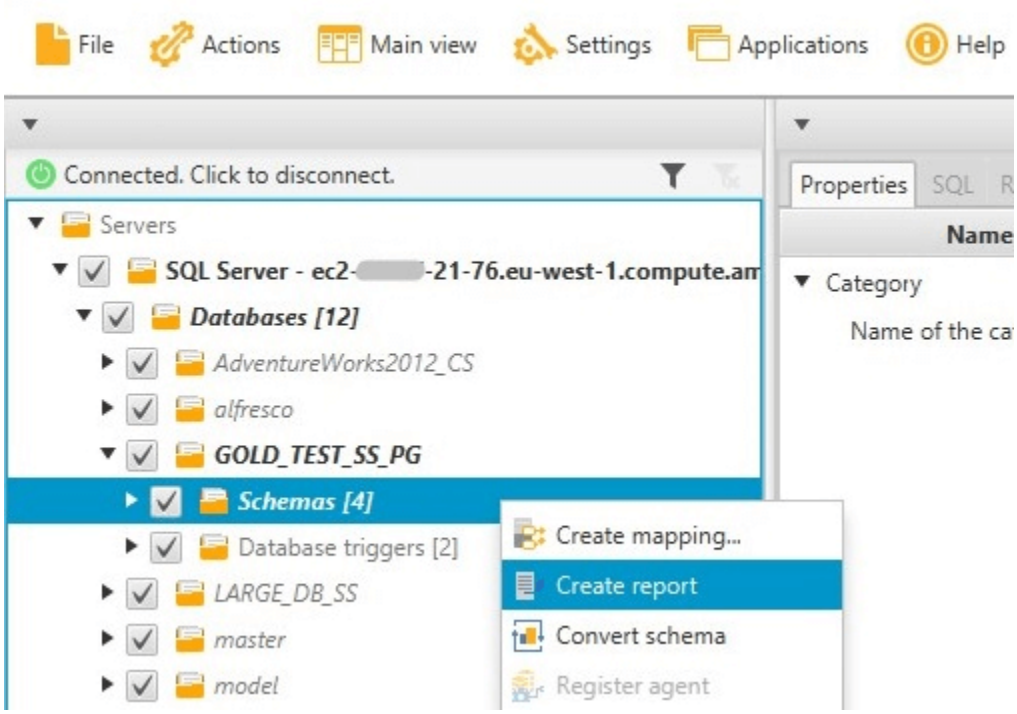
データベース移行評価レポートの作成

次の手順に従ってデータベース移行評価レポートを作成します。

データベース移行評価レポートを作成するには

1. 評価レポートを作成するソースデータベーススキーマのマッピングルールを作成していることを確認してください。詳細については、「新しいマッピングルールの追加」を参照してください。
2. [ビュー] メニューで、[メインビュー] を選択します。
3. ソースデータベースのスキーマを表示する左のパネルで、評価レポートを作成するスキーマオブジェクトを選択します。レポートに複数のデータベーススキーマを含める必要がある場合は、例えば [スキーマ] などの親ノードを選択します。

   評価レポートを作成するすべてのスキーマオブジェクトのチェックボックスがオンになっていることを確認してください。
4. オブジェクトのコンテキスト (右クリック) メニューを開き、[レポートを作成] を選択します。



評価レポートを表示する

評価レポートを作成すると、評価レポートビューが開き、次のタブが表示されます。

- 概要
- アクション項目

[Summary] (概要) タブには自動変換または未変換の項目が表示されます。

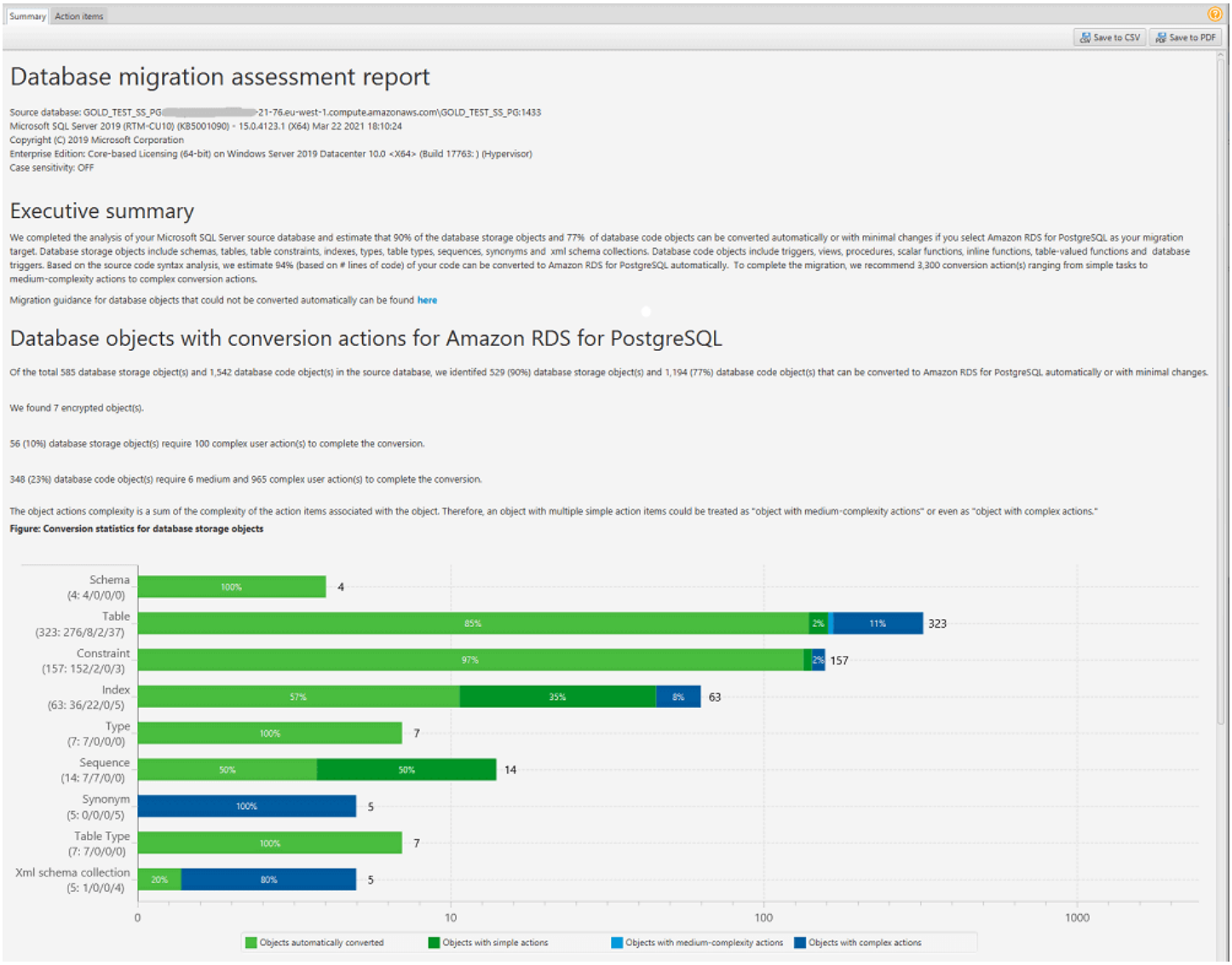
[Action Items] (アクション項目) タブには自動的に変換できなかった項目とその対処方法についてのアドバイスが表示されます。

トピック

- 評価レポートの概要
- 評価レポートのアクション項目
- 評価レポートの警告メッセージ

評価レポートの概要

[Summary] (概要) タブには、データベース移行評価レポートの要約が表示されます。自動的に変換された項目と、自動的に変換されなかった項目が表示されます。

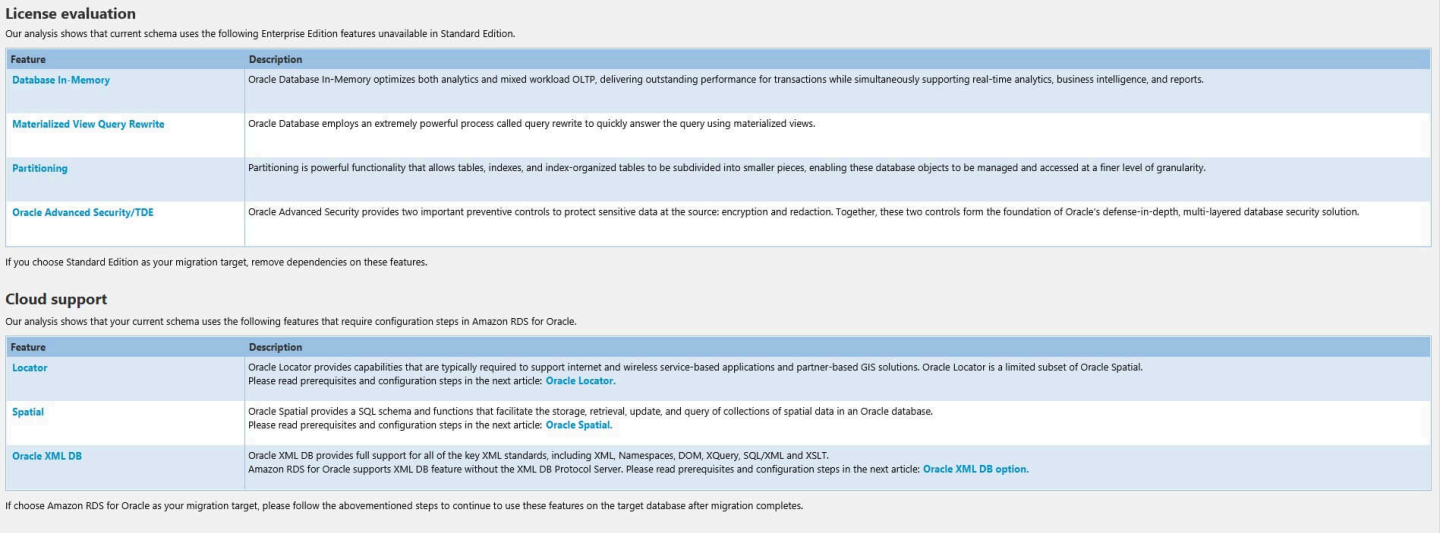


ターゲットデータベースエンジンに自動的に変換できないスキーマ項目に関して、要約には、ソースと同等のスキーマ項目をターゲット DB インスタンスで作成するために必要な労力の予測が含まれます。

このレポートは、これらのスキーマアイテムを変換する推定時間を次のように分類します。

- [Simple] (シンプル) – 2 時間以内に完了できるアクション。
- [Medium] (ミディアム) – より複雑で、2 ~ 6 時間で完了できるアクション。
- [Significant] (大規模) – 非常に複雑で、完了に 6 時間以上かかるアクション。

[License Evaluation and Cloud Support] (ライセンス評価およびクラウドサポート) セクションには、同エンジンを実行する Amazon RDS DB インスタンスへの既存のオンプレミスのデータベーススキーマの移動に関する情報が含まれます。例えば、ライセンスのタイプを変更する場合は、このレポートセクションを確認することで、現在のデータベースから削除する必要のある機能を見極めることができます。



License evaluation

Our analysis shows that current schema uses the following Enterprise Edition features unavailable in Standard Edition.

Feature	Description
Database In-Memory	Oracle Database In-Memory optimizes both analytics and mixed workload OLTP, delivering outstanding performance for transactions while simultaneously supporting real-time analytics, business intelligence, and reports.
Materialized View Query Rewrite	Oracle Database employs an extremely powerful process called query rewrite to quickly answer the query using materialized views.
Partitioning	Partitioning is powerful functionality that allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity.
Oracle Advanced Security/TDE	Oracle Advanced Security provides two important preventive controls to protect sensitive data at the source: encryption and redaction. Together, these two controls form the foundation of Oracle's defense-in-depth, multi-layered database security solution.

If you choose Standard Edition as your migration target, remove dependencies on these features.

Cloud support

Our analysis shows that your current schema uses the following features that require configuration steps in Amazon RDS for Oracle.

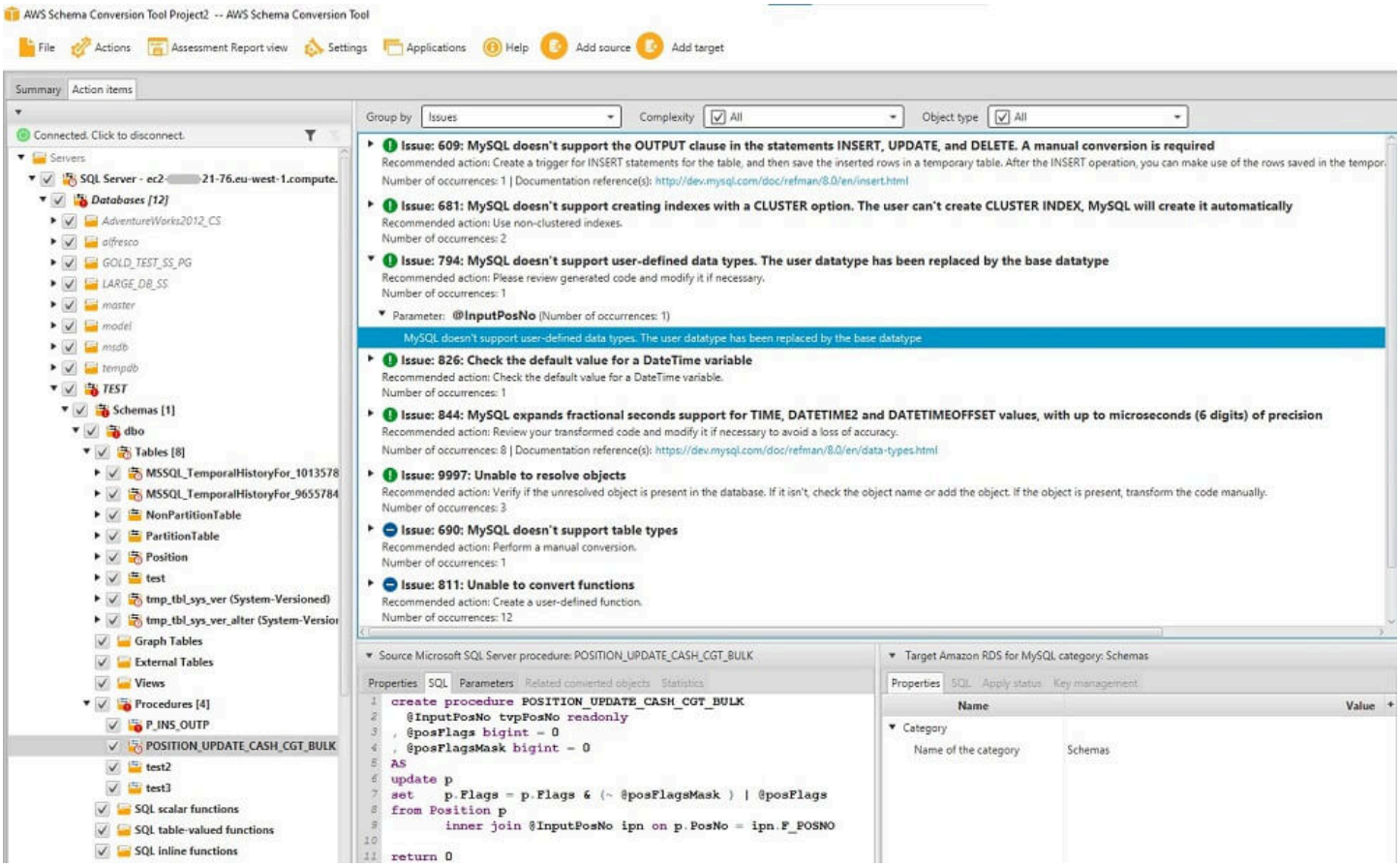
Feature	Description
Locator	Oracle Locator provides capabilities that are typically required to support internet and wireless service-based applications and partner-based GIS solutions. Oracle Locator is a limited subset of Oracle Spatial. Please read prerequisites and configuration steps in the next article: Oracle Locator.
Spatial	Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial data in an Oracle database. Please read prerequisites and configuration steps in the next article: Oracle Spatial.
Oracle XML DB	Oracle XML DB provides full support for all of the key XML standards, including XML, Namespaces, DOM, XQuery, SQL/XML and XSLT. Amazon RDS for Oracle supports XML DB feature without the XML DB Protocol Server. Please read prerequisites and configuration steps in the next article: Oracle XML DB option.

If choose Amazon RDS for Oracle as your migration target, please follow the abovementioned steps to continue to use these features on the target database after migration completes.

評価レポートのアクション項目

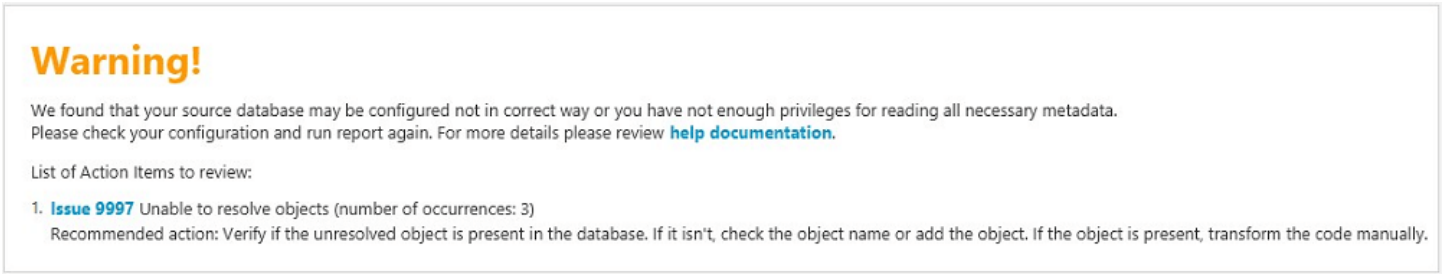
評価レポートビューには [Action Items] (アクション項目) タブも含まれます。このタブには、ターゲット Amazon RDS DB インスタンスのデータベースエンジンに自動的に変換できない項目のリストが含まれます。リストからアクション項目を選択する場合、アクション項目が適用されるスキーマが AWS SCT によってハイライトされます。

レポートには、手動でスキーマ項目を変換する方法の推奨事項も含まれています。例えば、評価の実行後のデータベース/スキーマの詳細レポートには、アクション項目の変換に関する推奨事項を設計および実装するために必要な労力が示されます。手動変換の処理方法の決定に関する詳細については、AWS SCT でのマニュアル変換の処理 を参照してください。



評価レポートの警告メッセージ

別のデータベースエンジンへの変換の複雑さを評価するには、AWS SCT がソースデータベース内のオブジェクトにアクセスできる必要があります。スキャン中に問題が発生したため SCT が評価を実行できない場合、全体的なコンバージョン率が低下したことを示す警告メッセージが表示されます。



次の理由により、スキャン中に AWS SCT に問題が発生する可能性があります。

- データベースに接続されているユーザーアカウントが、必要なすべてのオブジェクトにアクセスできない。
- スキーマで引用されたオブジェクトが、データベース内に存在しなくなった。
- SCT が、暗号化されたオブジェクトを評価しようとしています。

SCT で必要とされるデータベースのセキュリティ許可と権限の詳細については、「AWS SCT のソース」を参照して、このガイドの適切なソースデータベースのセクションを確認してください。

評価レポートの保存

データベース移行評価レポートの作成後、データベース移行評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらかとして保存できます。

データベース移行評価レポートを PDF ファイルとして保存する

1. トップメニューで、[ビュー] を選択してから、[評価レポートのビュー] を選択します。
2. [Summary] (概要) タブを選択します。
3. 右上の [Save to PDF] (PDF に保存) を選択します。

データベース移行評価レポートを CSV ファイルとして保存する

1. トップメニューで、[ビュー] を選択してから、[評価レポートのビュー] を選択します。
2. [Summary] (概要) タブを選択します。
3. 右上の [Save to CSV] (CSV に保存) を選択します。

PDF ファイルには、次の例に示すように、要約とアクション項目の情報の両方が含まれます。

Database objects with conversion actions for Amazon RDS for PostgreSQL

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identifed 529 (90%) database storage object(s) and 1,194 (77%) database code object(s) that can be converted to Amazon RDS for PostgreSQL automatically or with minimal changes.

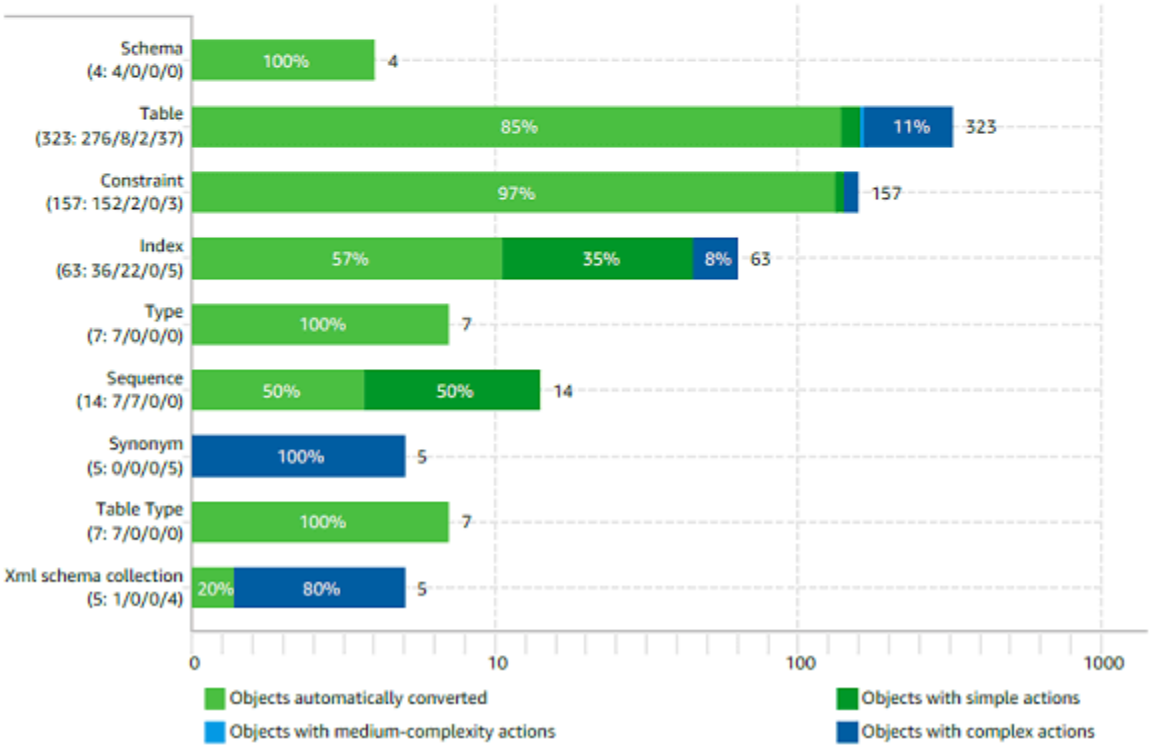
We found 7 encrypted object(s).

56 (10%) database storage object(s) require 100 complex user action(s) to complete the conversion.

348 (23%) database code object(s) require 6 medium and 965 complex user action(s) to complete the conversion.

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Figure: Conversion statistics for database storage objects



[CSV に保存] オプションを選択すると、AWS SCT で 3 つの CSV ファイルが作成されます。

最初の CSV ファイルには、アクション項目に関する次の情報が含まれています。

- カテゴリ
- オカレンス — ファイル名、行番号、および商品の位置
- アクション項目番号
- 件名
- グループ

- 説明
- ドキュメントリファレンス
- 推奨されるアクション
- 推定される複雑度

2 番目の CSV ファイルには、その名前に Action_Items_Summary サフィックスが含まれ、すべてのアクション項目のオカレンスに関する情報が含まれます。

次の例では、[Learning curve effort] (学習曲線の労力) 列の値は、各アクション項目を変換するアプローチを設計するために必要な労力を示します。[Effort to convert an occurrence of the action item] (アクション項目のオカレンスを変換する努力) 列の値は、設計されたアプローチに従って、各アクション項目の変換に必要な労力を示します。必要な労力のレベルを示すために使用される値は、低 (最小) から高 (最高) までの加重スケールに基づいています。

Schema	Action item	Number of occurrences	Learning curve efforts	Efforts to convert an occurrence of the action item
TEST.dbo	609	1	8	0.3
TEST.dbo	681	2	0.1	0.1
TEST.dbo	690	1	40	40
TEST.dbo	794	1	0	0.01
TEST.dbo	811	12	40	8
TEST.dbo	826	1	0	0.1
TEST.dbo	844	8	8	0.5
TEST.dbo	9997	3	0	0.3

3 番目の CSV ファイルには、その名前に Summary が含まれ、次の要約が含まれています。

- カテゴリ
- プロジェクト数
- 自動的に変換されるオブジェクト
- 単純なアクションのあるオブジェクト
- 複雑度が中程度のアクションのあるオブジェクト
- 複雑なアクションのあるオブジェクト
- 総コード行数

評価レポートを設定する

AWS SCT で評価レポートに含める詳細の量は構成できます。

データベース移行評価レポートを設定するには

1. [設定] メニューで [グローバル設定] を選択し、[評価レポート] を選択します。
2. [アクションアイテムの発生回数] では、[最初の 5 つの問題のみ] を選択して、評価レポートに含まれる 1 種類のアクションの商品の数を制限します。評価レポートに各タイプのアクションアイテムをすべて含めるには、[すべての問題] を選択します。
3. [SQL スクリプトで分析されたファイル] の場合は、[***X*** ファイル以下を一覧表示] を選択して、*X* への評価レポートに含まれる SQL スクリプトファイルの数を制限します。ファイルの数を入力します。評価レポートにすべての SQL スクリプトファイルを含めるには、[分析されたすべてのファイルを一覧表示] を選択します。
4. [保存後にレポートを開く] を選択すると、データベース移行評価レポートのローカルコピーを保存した後にファイルが自動的に開きます。詳細については、「

データベース移行評価レポートの作成後、データベース移行評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらかとして保存できます。

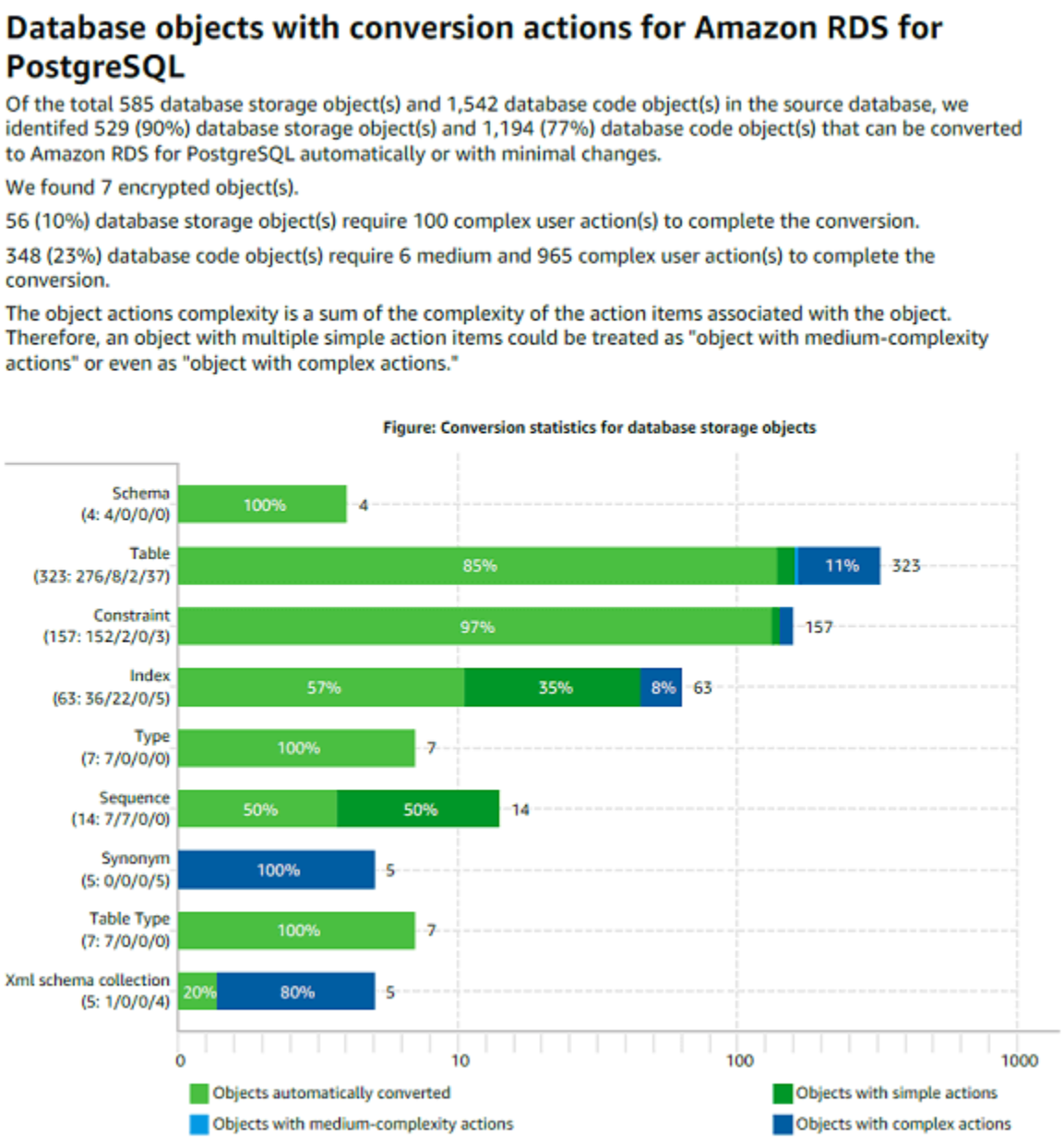
データベース移行評価レポートを PDF ファイルとして保存する

1. トップメニューで、[ビュー] を選択してから、[評価レポートのビュー] を選択します。
2. [Summary] (概要) タブを選択します。
3. 右上の [Save to PDF] (PDF に保存) を選択します。

データベース移行評価レポートを CSV ファイルとして保存する

1. トップメニューで、[ビュー] を選択してから、[評価レポートのビュー] を選択します。
2. [Summary] (概要) タブを選択します。
3. 右上の [Save to CSV] (CSV に保存) を選択します。

PDF ファイルには、次の例に示すように、要約とアクション項目の情報の両方が含まれます。



[CSV に保存] オプションを選択すると、AWS SCT で 3 つの CSV ファイルが作成されます。

最初の CSV ファイルには、アクション項目に関する次の情報が含まれています。

- カテゴリ
- オカレンス — ファイル名、行番号、および商品の位置
- アクション項目番号
- 件名

- グループ

- 説明
- ドキュメントリファレンス
- 推奨されるアクション
- 推定される複雑度

2 番目の CSV ファイルには、その名前に Action_Items_Summary サフィックスが含まれ、すべてのアクション項目のオカレンスに関する情報が含まれます。

次の例では、[Learning curve effort] (学習曲線の労力) 列の値は、各アクション項目を変換するアプローチを設計するために必要な労力を示します。[Effort to convert an occurrence of the action item] (アクション項目のオカレンスを変換する努力) 列の値は、設計されたアプローチに従って、各アクション項目の変換に必要な労力を示します。必要な労力のレベルを示すために使用される値は、低 (最小) から高 (最高) までの加重スケールに基づいています。

Schema	Action item	Number of occurrences	Learning curve efforts	Efforts to convert an occurrence of the action item
TEST.dbo	609	1	8	0.3
TEST.dbo	681	2	0.1	0.1
TEST.dbo	690	1	40	40
TEST.dbo	794	1	0	0.01
TEST.dbo	811	12	40	8
TEST.dbo	826	1	0	0.1
TEST.dbo	844	8	8	0.5
TEST.dbo	9997	3	0	0.3

3 番目の CSV ファイルには、その名前に Summary が含まれ、次の要約が含まれています。

- カテゴリ

- プロジェクト数
- 自動的に変換されるオブジェクト
- 単純なアクションのあるオブジェクト
- 複雑度が中程度のアクションのあるオブジェクト
- 複雑なアクションのあるオブジェクト
- 総コード行数

」を参照してください。

データベース移行評価レポートの作成

環境全体にとって最適なターゲットの方向性を決定するには、マルチサーバー評価レポートを作成します。

マルチサーバー評価レポートは、評価する各スキーマの定義に対して指定した入力に基づいて、複数のサーバーを評価します。スキーマ定義には、データベースサーバーの接続パラメータと各スキーマのフルネームが含まれています。各スキーマを評価した後、AWS SCT は、複数のサーバーにわたるデータベース移行に関する概要をまとめた評価レポートを作成します。このレポートには、移行ターゲットごとに推定される複雑さが表示されます。

AWS SCT を使用して、以下のソースデータベースとターゲットデータベースに関するマルチサーバー評価レポートを作成できます。

ソースデータベース	ターゲットデータベース
Amazon Redshift	Amazon Redshift
Azure SQL データベース	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
Azure Synapse Analytics	Amazon Redshift
BigQuery	Amazon Redshift
Greenplum	Amazon Redshift
IBM Db2 for z/OS	Amazon Aurora MySQL 互換エディション (Aurora MySQL)、Amazon Aurora PostgreSQL 互換エディション (Aurora PostgreSQL)、MySQL, PostgreSQL
IBM Db2 LUW	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL
Microsoft SQL Server	Aurora MySQL、Aurora PostgreSQL、Amazon Redshift、Babelfish for Aurora PostgreSQL、MariaDB、Microsoft SQL Server、MySQL、PostgreSQL

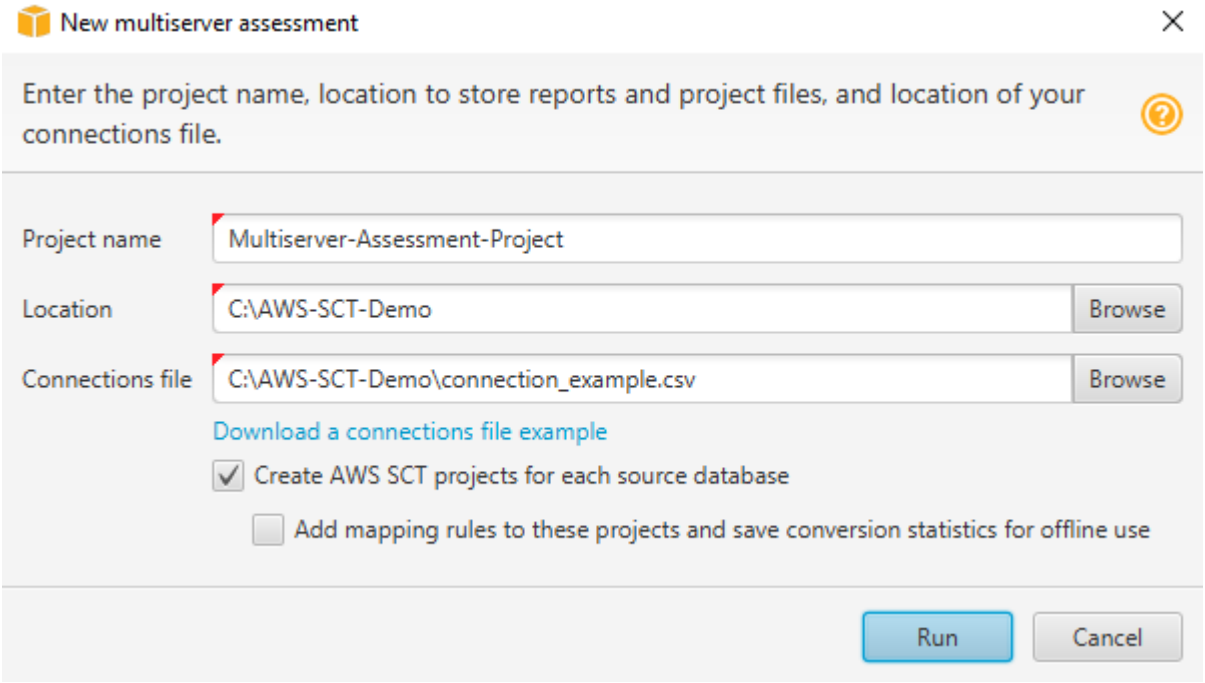
ソースデータベース	ターゲットデータベース
MySQL	Aurora PostgreSQL、MySQL、PostgreSQL
Netezza	Amazon Redshift
Oracle	Aurora MySQL、Aurora PostgreSQL、Amazon Redshift、MariaDB、MySQL、Oracle、PostgreSQL
PostgreSQL	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
SAP ASE	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL
Snowflake	Amazon Redshift
Teradata	Amazon Redshift
Vertica	Amazon Redshift

マルチサーバー評価を実行する

以下の手順を使用して、AWS SCT でマルチサーバー評価を実行します。マルチサーバー評価を実行するのに、AWS SCT で新しいプロジェクトを作成する必要はありません。開始する前に、データベース接続パラメータを含むカンマ区切り値 (CSV) ファイルを準備していることを確認してください。また、必要なデータベースドライバーがすべてインストールされていることを確認し、AWS SCT の設定でドライバーの場所を設定してください。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。

マルチサーバー評価を実行し、集約されたサマリーレポートを作成するには

1. AWS SCT で、[File] (ファイル)、[New multiserver assessment] (新しいマルチサーバー評価) を選択します。[New multiserver assessment] (新しいマルチサーバー評価) ダイアログボックスが開きます。



2. データベース接続パラメーターを含む CSV ファイルの空のテンプレートをダウンロードするには、「接続ファイルの例をダウンロード」を選択します。
3. [プロジェクト名]、[ロケーション] (レポートの保存先)、および [接続ファイル] (.csv ファイル) の値を入力します。
4. 評価レポートの生成後に移行プロジェクトを自動的に作成するには、[ソースデータベースごとに AWS SCT プロジェクトを作成] を選択します。
5. [ソースデータベースごとに AWS SCT プロジェクトを作成] をオンにすると、[これらのプロジェクトにマッピングルールを追加し、変換統計をオフラインで使用できるように保存する] を選択できます。この場合、AWS SCT は各プロジェクトにマッピングルールを追加し、ソースデータベースのメタデータをプロジェクトに保存します。詳細については、「AWS SCT のオフラインモードでの実行」を参照してください。
6. [実行] を選択します。

   データベース評価のペースを示す進行状況バーが表示されます。ターゲットエンジンの数は、評価のランタイムに影響を与える可能性があります。
7. 「すべてのデータベースサーバーの完全な分析には時間がかかる場合があります」というメッセージが表示された場合は、[はい] を選択します。続行しますか?

   マルチサーバー評価レポートが完了すると、そのことを示す画面が表示されます。
8. [レポートを開く] を選択して、集約されたサマリー評価レポートを表示します。

デフォルトでは、AWS SCT はすべてのソースデータベースの集約レポートと、ソースデータベースの各スキーマ名ごとの、詳細な評価レポートを生成します。詳細については、「レポートを検索して表示する」を参照してください。

[ソースデータベースごとに AWS SCT プロジェクトを作成] AWS SCT オプションをオンにすると、ソースデータベースごとに空のプロジェクトが作成されます。 AWS SCT は、前述のように評価レポートも作成します。これらの評価レポートを分析し、ソースデータベースごとに移行先を選択したら、これらの空のプロジェクトにターゲットデータベースを追加します。

[これらのプロジェクトにマッピングルールを追加し、変換統計をオフラインで使用できるように保存する] オプションをオンにした状態で、AWS SCT はソースデータベースごとにプロジェクトを作成します。これらのプロジェクトには、次の情報が含まれます。

- ソースデータベースと仮想ターゲットデータベースプラットフォーム。詳細については、「仮想ターゲットの使用」を参照してください。
- このソースとターゲットのペアのマッピングルール。詳細については、「マッピングルールの作成」を参照してください。
- このソースとターゲットのペアのデータベース移行評価レポート。
- ソーススキーマのメタデータ。これにより、この AWS SCT プロジェクトをオフラインモードで使用できるようになります。詳細については、「AWS SCT のオフラインモードでの実行」を参照してください。

入力 CSV ファイルの準備

接続パラメータをマルチサーバー評価レポートの入力として指定するには、次の例に示すように、CSV ファイルを使用します。

```
Name,Description,Secret Manager Key,Server IP,Port,Service Name,Database name,BigQuery
 path,Source Engine,Schema Names,Use Windows Authentication,Login,Password,Use
 SSL,Trust store,Key store,SSL authentication,Target Engines
Sales,,,192.0.2.0,1521,pdb,,,ORACLE,Q4_2021;FY_2021,,user,password,,,,,POSTGRESQL;AURORA_POSTGR
Marketing,,,ec2-a-b-c-d.eu-
west-1.compute.amazonaws.com,1433,,target_audience,,MSSQL,customers.dbo,,user,password,,,,,AURO
HR,,,192.0.2.0,1433,,employees,,MSSQL,employees.%,true,,,,,,,AURORA_POSTGRESQL
Customers,,secret-name,,,,,,MYSQL,customers,,,,,,,,AURORA_POSTGRESQL
Analytics,,,198.51.100.0,8195,,STATISTICS,,DB2LUW,BI_REPORTS,,user,password,,,,,POSTGRESQL
Products,,,203.0.113.0,8194,,,,TERADATA,new_products,,user,password,,,,,REDSHIFT
```

前の例では、セミコロンを使用して Sales データベースの 2 つのスキーマ名を区切っています。また、セミコロンを使用して Sales データベースの 2 つのターゲットデータベース移行プラットフォームを区切っています。

また、前の例では AWS Secrets Manager を使用して Customers データベースに接続し、Windows 認証を使用して HR データベースに接続します。

新しい CSV ファイルを作成するか、AWS SCT から CSV ファイルのテンプレートをダウンロードして、必要な情報を入力できます。CSV ファイルの最初の行に、前の例に示したのと同じ列名が含まれていることを確認してください。

入力 CSV ファイルのテンプレートをダウンロードするには

1. AWS SCT を起動します。
2. [ファイル]、[新しいマルチサーバー評価] の順に選択します。
3. [接続ファイルのサンプルをダウンロード] を選択します。

CSV ファイルに、テンプレートで指定された次の値が含まれていることを確認してください。

- [名前] — データベースの識別に役立つテキストラベル。 AWS SCT は、このテキストラベルを評価レポートに表示します。
- [説明] — オプションの値で、データベースに関する追加情報を入力できます。
- [シークレットマネージャーキー] — データベースの認証情報を AWS Secrets Manager に保存するシークレットの名前。Secrets Manager を使用するには、必ず AWS プロファイルを AWS SCT に保存してください。詳細については、「AWS Secrets Manager を使用する」を参照してください。

> **Important**
>
> 入力ファイルに [サーバー IP]、[ポート]、[ログイン]、および [パスワード] のパラメーターを含めた場合、AWS SCT は、[Secret Manager キー] パラメーターを無視します。

- [サーバー IP] – ソースデータベースサーバーのドメインネームサービス (DNS) 名または IP アドレスを入力します。
- [ポート]: ソースデータベースサーバーへの接続に使用するポート。
- [サービス名] — サービス名を使用して Oracle データベースに接続する場合、接続する Oracle サービスの名前です。

- [データベース名] — データベース名。Oracle データベースの場合は、Oracle システム ID (SID) を使用します。
- [BigQuery パス] — ソース BigQuery データベースのサービスアカウントキーファイルへのパス。このファイルの作成の詳細については、「BigQuery をソースとする場合の権限」を参照してください。
- [ソースエンジン] — ソースデータベースのタイプ。次のいずれかの値を使用します。
  - Azure SQL データベース用の [AZURE_MSSQL]。
  - Azure Synapse Analytics データベース用の [AZURE_SYNAPSE]。
  - BigQuery データベース用の [GOOGLE_BIGQUERY]。
  - IBM Db2 for z/OS データベース用の [DB2ZOS]。
  - IBM Db2 LUW データベース用の [DB2LUW]。
  - Greenplum データベース用の [GREENPLUM]。
  - Microsoft SQL Server データベース用の [MSSQL]。
  - MySQL データベース用の [MYSQL]。
  - Netezza データベース用の [NETEZZA]。
  - Oracle データベース用の [ORACLE]。
  - PostgreSQL データベース用の [POSTGRESQL]。
  - Amazon Redshift データベースの [REDSHIFT]。
  - Snowflake データベース用の [SNOWFLAKE]。
  - SAP ASE データベース用の [SYBASE_ASE]。
  - Teradata データベース用の [TERADATA]。
  - VERTICA はVertica データベース用の [VERTICA]。
- [スキーマ名] — 評価レポートに含めるデータベーススキーマの名前。

  Azure SQL Database、Azure Synapse Analytics、BigQuery、Netezza、SAP ASE、Snowflake、SQL Server では、次の形式のスキーマ名を使用してください。

  db_name.*schema_name*

  db_name をソースデータベースの名前に置き換えます。

  schema_name をソーススキーマの名前に置き換えます。

ドットを含むデータベース名またはスキーマ名は、次に示すように二重引用符で囲みます: `"database.name"."schema.name"`。

次に示すように、複数のスキーマ名をセミコロンで区切ります: `Schema1;Schema2`。

データベース名とスキーマ名では大文字が区別されます。

パーセント (%) をワイルドカードとして使用すると、データベース名またはスキーマ名に含まれる任意の数のシンボルを置き換えることができます。前の例では、パーセント (%) をワイルドカードとして使用して、`employees` データベースのすべてのスキーマを評価レポートに含めています。

- [Windows 認証を使用する] — Windows 認証を使用して Microsoft SQL Server データベースに接続する場合は、[true] と入力します。詳細については、「Microsoft SQL Server をソースとして使用するときの Windows 認証の使用」を参照してください。
- [ログイン] – ソース データベース サーバーに接続するためのユーザー名。
- [パスワード] – パスワードを入力して、ソースデータベースサーバーに接続します。
- [SSL を使用する] — ソースデータベースへの接続に Secure Sockets Layer (SSL) を使用する場合は、true を入力します。
- [信頼ストア] — SSL 接続に使用する信頼ストア。
- [キーストア] — SSL 接続に使用するキーストア。
- [SSL 認証] — 証明書による SSL 認証を使用する場合は、true を入力します。
- [ターゲットエンジン] — ターゲットデータベースプラットフォーム。以下の値を使用して、評価レポートで 1 つ以上のターゲットを指定します。
  - Aurora MySQL 互換データベース用の [AURORA_MYSQL] 。
  - Aurora PostgreSQL 互換データベース用の [AURORA_POSTGRESQL]。
  - Babelfish for Aurora PostgreSQL データベース用の [BABELFISH]。
  - MariaDB データベース用の [MARIA_DB]。
  - Microsoft SQL Server データベース用の [MSSQL]。
  - MySQL データベース用の [MYSQL]。
  - Oracle データベース用の [ORACLE]。
  - PostgreSQL データベース用の [POSTGRESQL]。
  - Amazon Redshift データベースの [REDSHIFT]。

  複数のターゲットは、次のようにセミコロンを使用して区切ります: `MYSQL;MARIA_DB` ターゲットの数は、評価の実行にかかる時間に影響します。

レポートを検索して表示する

マルチサーバー評価では、2 種類のレポートが生成されます。

- すべてのソースデータベースの集約レポート。
- ソースデータベースの各スキーマ名ごとの、ターゲットデータベースに関する詳細な評価レポート。

レポートは、[新しいマルチサーバー評価] ダイアログボックス内の [ロケーション] で選択したディレクトリに保存されます。

詳細レポートにアクセスするには、ソースデータベース、スキーマ名、ターゲットのデータベースエンジン別に編成されたサブディレクトリをナビゲートします。

集約されたレポートでは、ターゲットデータベースの変換の複雑さに関する情報が 4 列にわたり表示されます。列には、コードオブジェクトの変換、ストレージオブジェクト、構文要素、および変換の複雑さに関する情報が含まれます。

次の例は、2 つの Oracle データベーススキーマを Amazon RDS for PostgreSQL に変換するための情報を示しています。

Server IP address and port	Secret Manager key	Name	Description	Database name	Schema name	Code object conversion % for "Amazon RDS for PostgreSQL"	Storage object conversion % for "Amazon RDS for PostgreSQL"	Syntax Elements conversion % for "Amazon RDS for PostgreSQL"	Conversion Complexity for "Amazon RDS for PostgreSQL"
192.0.2.0:1521		Sales		ORCL	Q4_2021	97.78%	100.00%	98.76%	1
192.0.2.0:1521		Sales		pdb	FY_2021	82.35%	85.19%	99.24%	10

指定した追加のターゲットデータベースエンジンごとに、同じ 4 つの列がレポートに追加されます。

この情報の読み方の詳細については、次を参照してください。

集約評価レポートを出力する

AWS Schema Conversion Tool の集計されたマルチサーバーデータベース移行評価レポートは、次の列を含む CSV ファイルです。

- `Server IP address and port`
- `Secret Manager key`
- `Name`
- `Description`

- Database name
- Schema name
- Code object conversion % for *target_database*
- Storage object conversion % for *target_database*
- Syntax elements conversion % for *target_database*
- Conversion complexity for *target_database*

情報を収集するために、AWS SCT は完全な評価レポートを実行し、スキーマ別にレポートを集約します。

レポートでは、次の 3 つのフィールドに、評価に基づいて可能な自動コンバージョンの割合が表示されます。

コードオブジェクトの変換率

AWS SCT が自動的に、または最小限の変更で変換できるスキーマ内のコード オブジェクトの割合です。コードオブジェクトには、プロシージャ、関数、ビューなどが含まれます。

ストレージオブジェクトの変換率

SCT が自動的に、または最小限の変更で変換できるストレージオブジェクトの割合を示します。ストレージオブジェクトには、テーブル、インデックス、制約などが含まれます。

構文要素の変換率

SCT が自動的に変換できる構文要素の割合を示します。構文要素には SELECT、FROM、DELETE、JOIN 句などが含まれます。

変換の複雑さの計算は、アクション項目の概念に基づいています。アクション項目は、特定のターゲットへの移行中に手動で修正する必要があるソースコード内の問題のタイプを反映しています。アクション項目には複数のオカレンスがある場合があります。

加重スケールは、移行を実行する際の複雑さのレベルを示します。数字の 1 は最低レベルの複雑さを表し、10 は最高レベルの複雑さを表します。

AWS SCT を使用したデータベーススキーマの変換

AWS Schema Conversion Tool (AWS SCT) を使用すると、データベースエンジン間で既存のデータベーススキーマを変換できます。AWS SCT ユーザーインターフェイスを使用したデータベースの変換は比較的簡単ですが、変換を行う前に、いくつかの点について考慮する必要があります。

例えば、AWS SCT を使用して以下を実行できます。

- AWS SCT を使用して、同エンジンを実行する Amazon RDS DB インスタンスに既存のオンプレミスのデータベーススキーマをコピーすることもできます。この機能を使用して、クラウドへの移動やライセンスタイプの変更にかかるコスト削減の可能性を分析できます。
- 場合によって、データベース機能は同等の Amazon RDS 機能に変換できないことがあります。Amazon Elastic Compute Cloud (Amazon EC2) プラットフォームでデータベースをホストして自己管理する場合は、これらの機能を AWS サービスで代用することで、エミュレートできます。
- AWS SCT はオンライントランザクション処理 (OLTP) データベーススキーマを Amazon Relational Database Service (Amazon RDS) MySQL DB インスタンス、Amazon Aurora DB クラスター、または PostgreSQL DB インスタンスに変換する多くのプロセスを自動化します。ソースおよびターゲットデータベースエンジンは様々な特徴と機能を含んでおり、AWS SCT は Amazon RDS DB インスタンスにおいて可能な限り同等のスキーマを作成するよう試みます。直接変換ができない場合、AWS SCT はユーザーに対して可能なアクションの一覧を提供します。

トピック

AWS SCT は、次のオンライントランザクション処理 (OLTP) 変換をサポートします。

ソースデータベース	ターゲットデータベース
IBM Db2 for z/OS (バージョン 12)	Amazon Aurora MySQL 互換エディション、Amazon Aurora PostgreSQL 互換エディション、MySQL、PostgreSQL
IBM Db2 LUW (バージョン 9.1、9.5、9.7、10.5、11.1、および 11.5)	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL
Microsoft Azure SQL データベース	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
Microsoft SQL Server (バージョン 2008 R2 以降)	Aurora MySQL、Aurora PostgreSQL、Babelfish for Aurora PostgreSQL、MariaDB、Microsoft SQL Server、MySQL、PostgreSQL
MySQL (バージョン 5.5 以上)	Aurora PostgreSQL、MySQL、PostgreSQL AWS SCT を使用せずに、MySQL から Aurora MySQL DB クラスターにスキーマとデータを移行できます。詳細については、「Amazon Aurora DB クラスターへのデータの移行」を参照してください。
Oracle (バージョン 10.2 以降)	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、Oracle、PostgreSQL
PostgreSQL (バージョン 9.1 以降)	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
SAP ASE (12.5、15.0、15.5、15.7、16.0)	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL

データウェアハウスの変換の詳細については、「AWS SCT を使用したデータウェアハウススキーマの Amazon Redshift への変換」を参照してください。

データベーススキーマを Amazon RDS に変換するには、次の必要な手順を実行します。

- AWS SCT での移行ルールの作成 - AWS SCT でスキーマを変換する前に、列のデータ型の変更、あるスキーマから別のスキーマへのオブジェクトの移動、オブジェクトの名前の変更を行うルールを設定できます。
- AWS SCT を使用してスキーマを変換する - AWS SCT は確認のために変換されたスキーマのローカルバージョンを作成しますが、準備ができるまでターゲット DB インスタンスには適用されません。
- AWS SCT を使用した移行評価レポートの作成 - AWS SCT 自動的に変換できないスキーマの要素の詳細を記載したデータベース移行評価レポートを作成します。ソースデータベースと互換性のある Amazon RDS DB インスタンスのどこでスキーマを作成する必要があるかを特定するため、このレポートを使用できます。
- AWS SCT でのマニュアル変換の処理 – 自動的に変換できないスキーマの要素がある場合、2 つの選択肢があります。ソーススキーマを更新したのち再び変換するか、またはターゲット Amazon RDS DB インスタンスにおいて同等のスキーマの要素を作成します。
- AWS SCT での変換されたスキーマの更新および再読み込み – ソースデータベースからの最新のスキーマを使用して AWS SCT プロジェクトを更新できます。
- AWS SCT で変換されたスキーマの保存および適用 - 準備ができたら、AWS SCT を使用してローカルプロジェクトにある変換後のスキーマをターゲット Amazon RDS DB インスタンスに適用します。

AWS SCT での移行ルールの作成

AWS SCT でスキーマを変換する前に、移行ルールを設定できます。AWS SCT の移行ルールでは、列のデータ型の変更、あるスキーマから別のスキーマへのオブジェクトの移動、オブジェクト名の変更などの変換を行うことができます。例えば、`test_TABLE_NAME` という名前のソーススキーマにテーブルがあるとします。ターゲットスキーマ内にある プレフィックス `test_` をプレフィックス `demo_` に変更するルールを設定できます。

> **Note**
>
> 移行ルールは、異なるソースとターゲットのデータベースエンジンに対してのみ作成できます。

移行ルールを作成して実行できるタスクは次のとおりです。

- プレフィックスの追加、削除、または置換

- サフィックスの追加、削除、または置換
- 列照合の変更
- データタイプの変更
- char、varchar、nvarchar、string およびデータ型の長さを変更
- オブジェクトの移動
- オブジェクトの名前変更

移行ルールを作成できるオブジェクトは次のとおりです。

- データベース
- スキーマ
- テーブル
- Column

移行ルールの作成

移行ルールを作成し、プロジェクトの一部として保存できます。プロジェクトを開いた状態で、以下の手順で移行ルールを作成します。

移行ルールを作成するには

1. [View] (ビュー) メニューで、[Mapping View] (マッピングビュー) を選択します。
2. [サーバーマッピング] で、ソースサーバとターゲットサーバのペアを選択します。
3. [New migration rule] (新しい移行ルール) を選択します。[Transformation rules] (変換ルール) ダイアログボックスが表示されます。
4. [Add new rule] (新しいルールを追加) を選択します。ルールのリストに新しい行が追加されます。
5. ルールを設定します。

    a. [Name] (名前) に、ルールの名前を入力します。

    b. [For] で、ルールを適用するオブジェクトのタイプを選択します。

    c. [where] (条件) に、オブジェクトに適用するフィルターを入力後、移行ルールを適用します。where 句は、like 句を使用して評価されます。正確な名前を入力して特定のオブジェクトを選択するか、パターンを入力して複数のオブジェクトを選択できます。

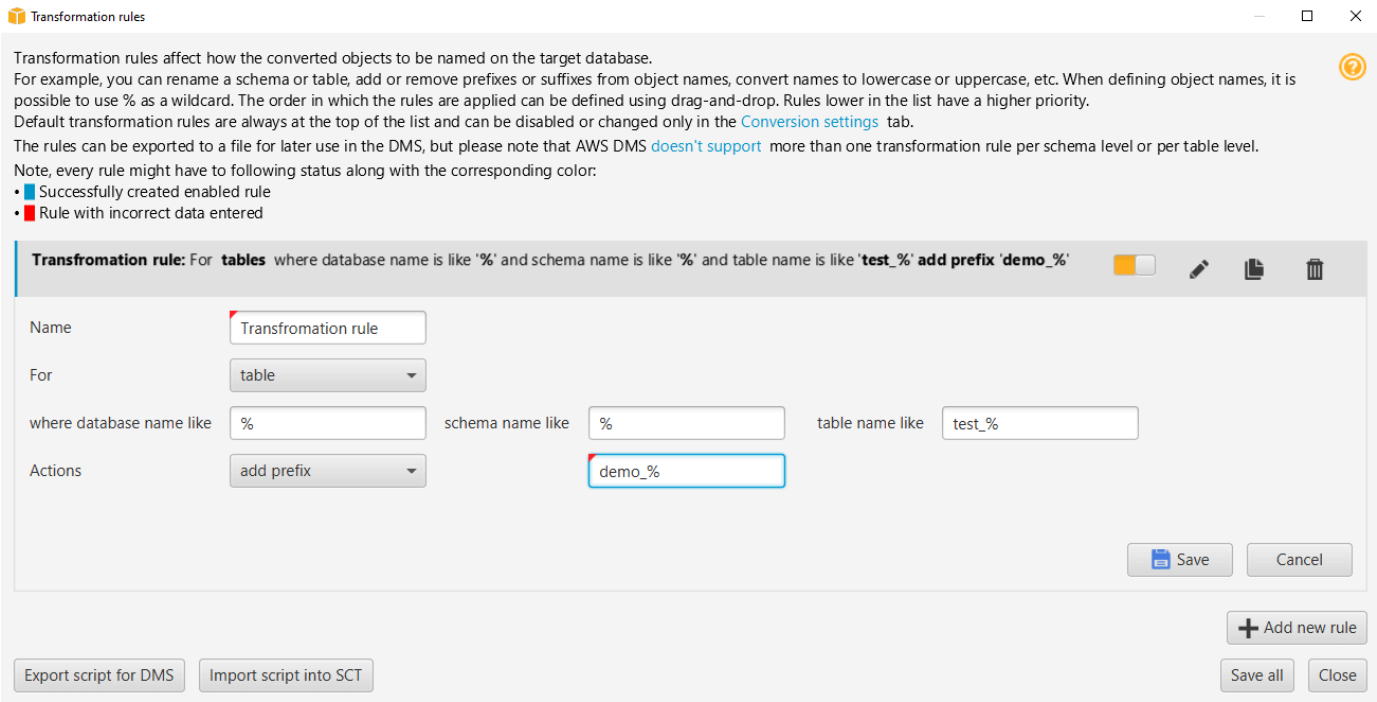
where 句で利用可能なフィールドは、オブジェクト型によって異なります。例えば、スキーマの名前のオブジェクト型がスキーマの場合、使用可能なフィールドは 1 つだけです。

d. [Actions] (アクション) で、作成する移行ルールを選択します。

e. ルールタイプに応じて、追加の値を 1 つまたは 2 つ入力します。例えば、オブジェクトの名前を変更するには、オブジェクトの新しい名前を入力します。プレフィックスを置換するには、現在のプレフィックスおよび置換後のプレフィックスを入力します。

   char、varchar、nvarchar、および文字列データ型の場合、乗算演算子を使用してデータ型の長さを変更できます。たとえば、`%*4` 値によって `varchar(10)` データ型が `varchar(40)` に変換されます。

6. 移行ルールを設定したら、[Save] (保存) を選択してルールを保存します。変更をキャンセルする場合は、[Cancel] (キャンセル) を選択します。



7. ルールの追加、編集、削除が完了したら、[Save All] (すべてを保存) を選択して変更内容を保存します。

8. [Close] (閉じる) を選択して、[Transformation rules] (変換ルール) ダイアログボックスを閉じます。

移行ルールを削除せずに無効にするには、切り替えアイコンを使用できます。既存の移行ルールを複製するには、コピーアイコンを使用します。既存の移行ルールを編集するには、鉛筆アイコンを使用

します。既存の移行ルールを削除するには、削除アイコンを使用します。移行ルールへの変更を保存するには、[Save All] (すべて保存) を選択します。

移行ルールのエクスポート

AWS DMS を使用してソースデータベースからターゲットデータベースにデータを移行する場合は、移行ルールに関する情報を AWS DMS に入力します。タスクに関する詳細は、「AWS Database Migration Service のレプリケーションタスクの使用」を参照してください。

移行ルールをエクスポートするには

1. AWS Schema Conversion Toolで、[View] (ビュー) メニューで [Mapping View] (マッピングビュー) を選択します。
2. [Migration rules] (移行ルール) で、移行ルールを選択してから、[Modify migration rule] (移行ルールの変更) を選択します。
3. [AWS DMS のスクリプトのエクスポート] を選択します。
4. スクリプトを保存する場所を指定し、[Save] (保存) を選択します。移行ルールは、AWS DMS で使用できる JSON スクリプトとして保存されます。

AWS SCT を使用してスキーマを変換する

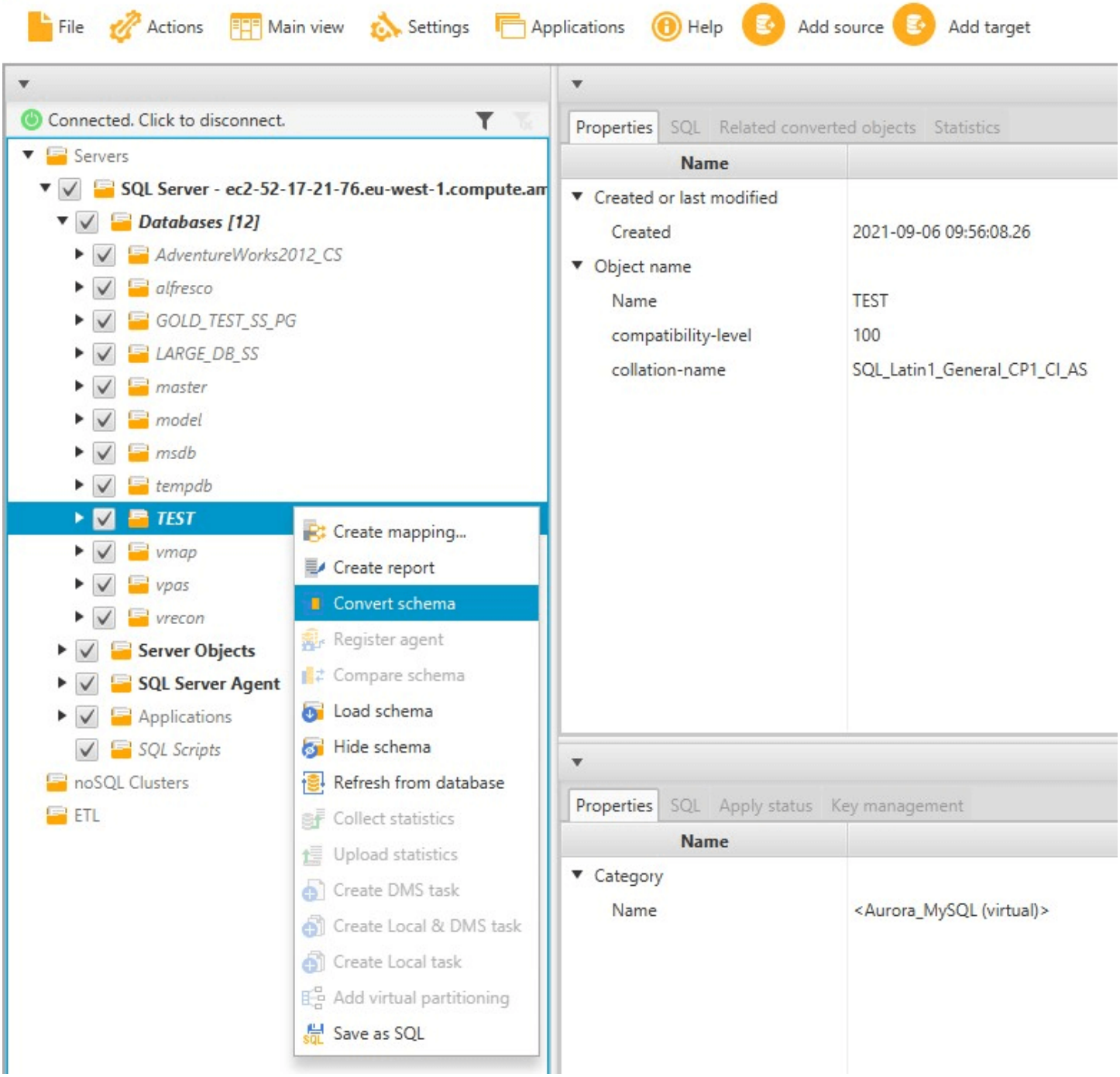
ソースデータベースおよびターゲット Amazon RDS DB インスタンスの両方にプロジェクトを接続したのち、AWS Schema Conversion Tool プロジェクトは左のパネルでソースデータベースからのスキーマを表示します。スキーマはツリービュー形式で表示され、ツリーの各ノードは遅延ロードです。ツリービューでノードを選択すると、その時点で AWS SCT はソースデータベースからスキーマ情報をリクエストします。

ソースデータベースからスキーマ項目を選択し、スキーマをターゲット DB インスタンスの DB エンジン向けに同等のスキーマに変換することができます。ソースデータベースから任意のスキーマ項目を選択して変換できます。選択したスキーマ項目が親項目に依存する場合、AWS SCT はその親項目にもスキーマを生成します。例えば、変換するテーブルを選択するとします。その場合は、AWS SCT ではテーブルのスキーマ、およびテーブルが格納されているデータベースが生成されます。

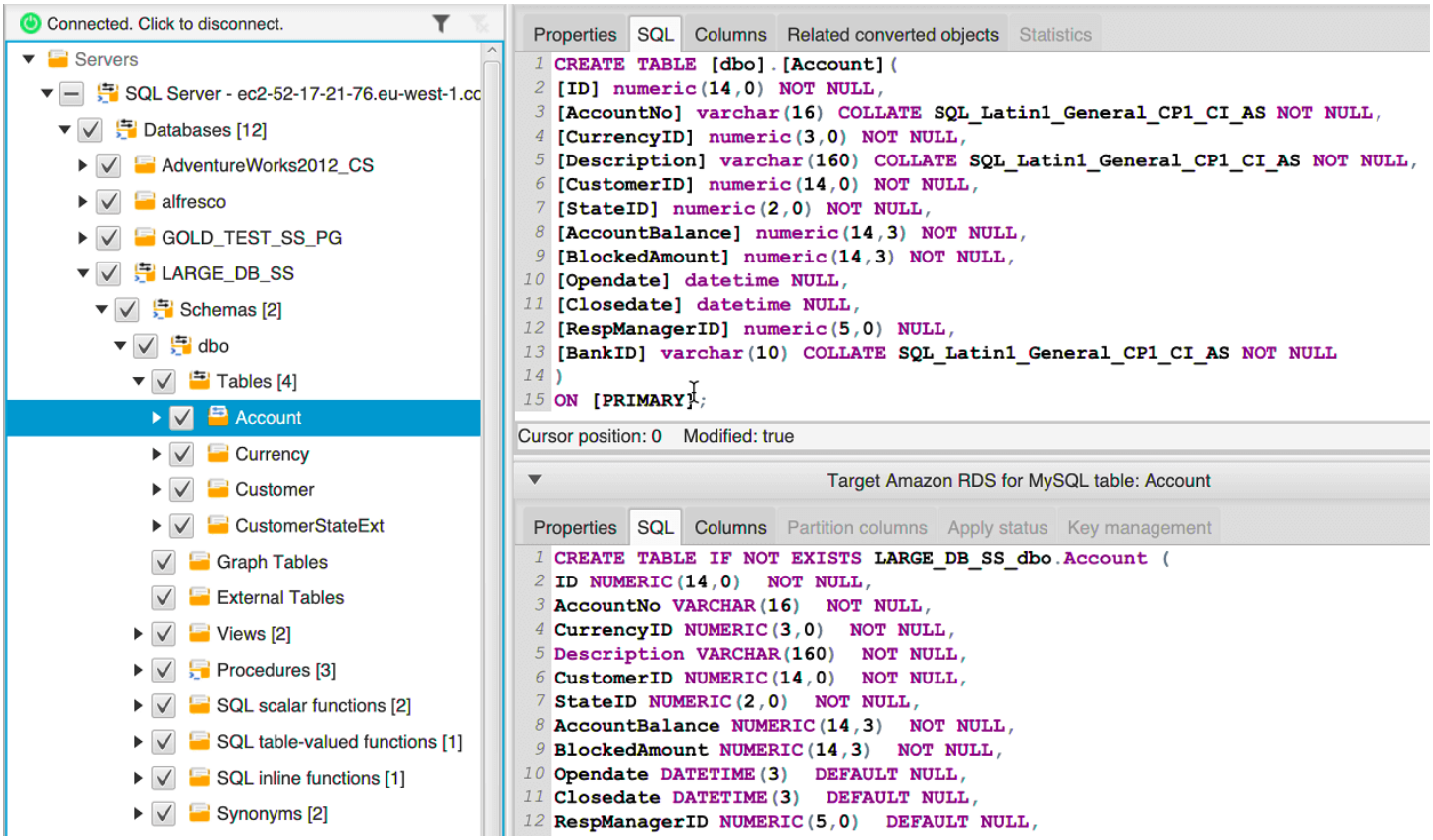
スキーマの変換

ソースデータベースからスキーマを変換するには、変換するスキーマの名前のチェックボックスを選択します。次に、プロジェクトの左側のパネルからこのスキーマを選択します。AWS SCT はスキー

マ名を青で強調表示します。スキーマのコンテキスト（右クリック）メニューを開き、次に示すように [スキーマの変換] を選択します。



ソースデータベースからスキーマを変換すると、プロジェクトの左のパネルからスキーマ項目を選択でき、プロジェクトの中央のパネルで変換されたスキーマを表示できます。下中央のパネルに変換されたスキーマを作成する SQL コマンドのプロパティが次のように表示されます。



スキーマを変換した後、プロジェクトを保存できます。ソースデータベースからのスキーマ情報は、プロジェクトと共に保存されます。この機能は、ソースデータベースに接続せずにオフラインで作業できることを意味します。ソースデータベースに [Refresh from Database] (データベースから更新) が選択されている場合、AWS SCT は必要に応じて、ソースデータベースに接続してプロジェクトのスキーマを更新します。詳細については、「AWS SCT での変換されたスキーマの更新および再読み込み」を参照してください。

自動的に変換できない項目のデータベース移行評価レポートを作成できます。評価レポートは、自動的に変換できないスキーマ項目の特定と解決に役立ちます。詳細については、「AWS SCT を使用した移行評価レポートの作成」を参照してください。

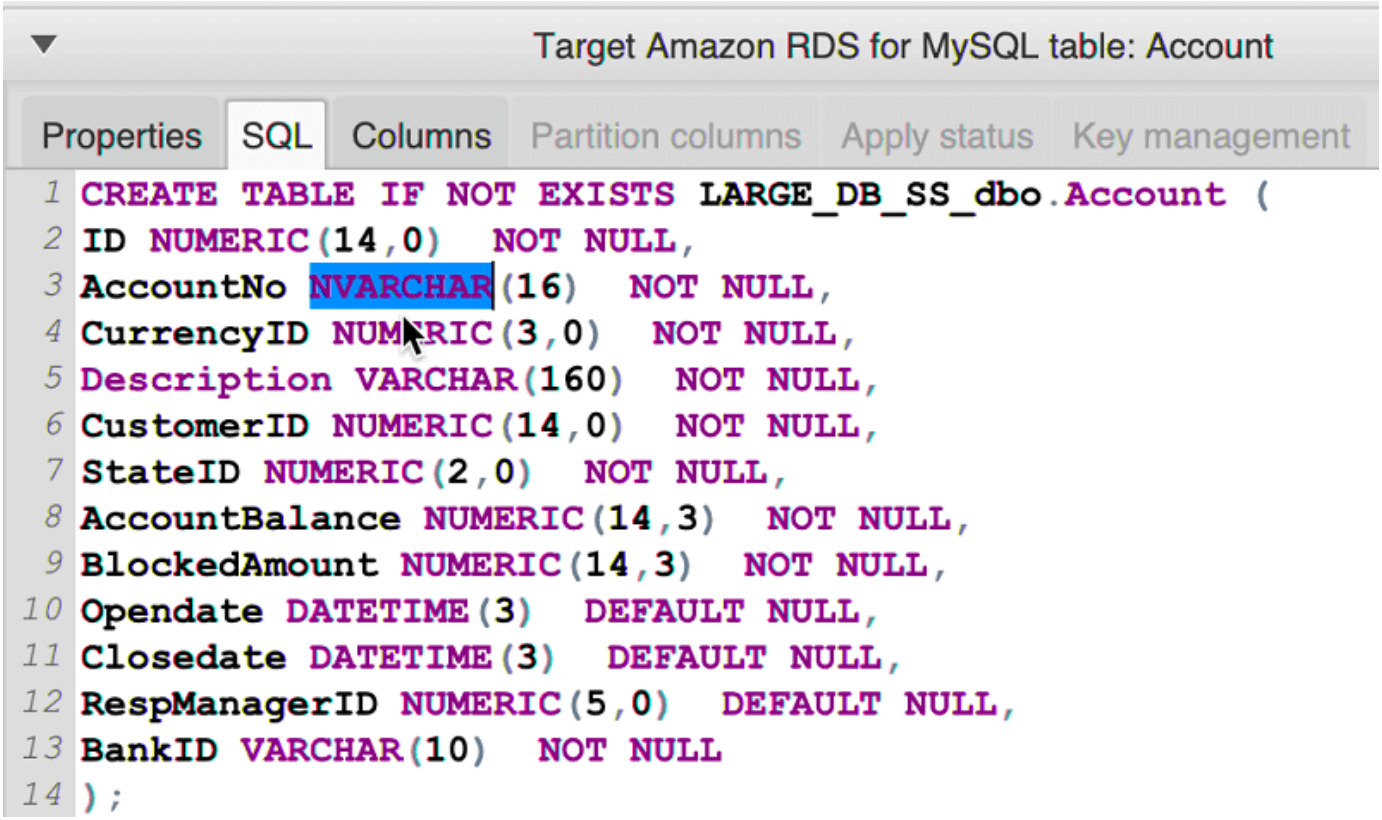
AWS SCT が変換されたスキーマを生成する場合、そのスキーマはターゲット DB インスタンスにすぐには適用されません。代わりに、変換されたスキーマはターゲット DB インスタンスに適用する準備ができるまでローカルに保存されます。詳細については、「変換されたスキーマの適用」を参照してください。

変換されたスキーマの編集

変換されたスキーマを編集し、プロジェクトの一部として変更を保存できます。

変換されたスキーマを編集するには

1. ソースデータベースのスキーマを表示する左のパネルで、変換されたスキーマの編集するスキーマ項目を選択します。
2. 選択した項目の変換されたスキーマを表示している下中央のパネルで [SQL] タブを選択します。
3. [SQL] タブで表示されたテキストで、必要に応じてスキーマを変更します。更新する際に、スキーマはプロジェクトと共に自動的に保存されます。



更新する際に、プロジェクトと共に変換されたスキーマへの変更が格納されます。新しくソースデータベースからスキーマ項目を変換し、項目の以前に変換されたスキーマを更新した場合、既存の更新はソースデータベースに基づいて新しく変換されたスキーマで置き換えられます。

変換されたスキーマのクリア

ターゲット DB インスタンスにスキーマを適用するまで、AWS SCT はプロジェクトで変換したスキーマをローカルでのみ保存します。DB インスタンスのツリービューノードを選択してから [Refresh from Database] (データベースから更新) を選択することにより、プロジェクトから計画されたスキーマをクリアできます。ターゲット DB インスタンスにスキーマが書き込まれていなかった

ため、データベースから更新することによって、AWS SCT プロジェクトで計画されたスキーマの要素は削除され、ソース DB インスタンスに存在するものと一致するようになります。

AWS SCT でのマニュアル変換の処理

評価レポートには、ターゲット Amazon RDS DB インスタンスのデータベースエンジンに自動的に変換できない項目のリストが含まれます。変換できない各項目に対して、[Action Items] (アクション項目) タブにアクション項目があります。

評価レポートのアクション項目には、以下の方法で対応できます。

- ソースデータベーススキーマの変更
- ターゲットデータベーススキーマの変更

ソーススキーマの変更

一部の項目については、ソースデータベースのデータベーススキーマを、自動的に変換できるスキーマに変更する方が容易な場合があります。最初に、新しい変更にアプリケーションアーキテクチャと互換性があることを確認し、次にソースデータベースのスキーマを更新します。最後に、更新されたスキーマ情報のあるプロジェクトを更新します。その後、更新されたスキーマを変換し、新しいデータベース移行評価レポートを生成できます。ソーススキーマで変更された項目のアクション項目は表示されなくなります。

このプロセスの利点は、ソースデータベースから更新するときに、更新されたスキーマを常に使用できることです。

ターゲットスキーマの変更

一部の項目については、ターゲットデータベースに変換されたスキーマを適用する方が容易な場合があるため、自動的に変換できなかった項目のターゲットデータベースに同等のスキーマを手動で追加します。スキーマを適用することで、ターゲット DB インスタンスに自動的に変換できるスキーマすべてを記述できます。詳細については、「AWS SCT で変換されたスキーマの保存および適用」を参照してください。

ターゲット DB インスタンスに書き込まれるスキーマには、自動的に変換できない項目は含まれません。ターゲット DB インスタンスにスキーマを適用したら、ターゲット DB インスタンスでソースデータベースと同等のスキーマを手動で作成できます。データベース移行評価レポートのアクション項目には、同等のスキーマを作成する方法の提案が含まれています。

> **Warning**
>
> ターゲット DB インスタンスでスキーマを手動で作成する場合、手動作業のコピーを保存します。再度プロジェクトからターゲット DB インスタンスに変換したスキーマを適用する場合、行った手動作業は上書きされます。

場合によっては、ターゲット DB インスタンスで同等のスキーマを作成できないことがあります。ターゲット DB インスタンス向けに DB エンジンから利用できる機能を使用するために、アプリケーションとデータベースの一部を再設計することが必要な場合があります。その他の場合、自動的に変換できないスキーマは無視できます。

AWS SCT での変換されたスキーマの更新および再読み込み

AWS Schema Conversion Tool プロジェクトのソーススキーマおよびターゲットスキーマの両方を更新できます。

- [Source] (ソース) - ソースデータベースのスキーマを更新した場合、AWS SCT はプロジェクトのスキーマをソースデータベースの最新のスキーマと置き換えます。この機能を使用すると、ソースデータベースのスキーマが変更された場合にプロジェクトを更新できます。
- [Target] (ターゲット) - ターゲット Amazon RDS DB インスタンスのスキーマを更新した場合、AWS SCT はプロジェクトのスキーマをターゲット DB インスタンスからの最新のスキーマと置き換えます。ターゲット DB インスタンスにスキーマを適用しなかった場合は、AWS SCT はプロジェクトから変換されたスキーマをクリアします。その後、クリーンターゲット DB インスタンスのソースデータベースからスキーマを変換します。

AWS SCT プロジェクトでスキーマを更新するには、[Refresh from Database] (データベースから更新) を選択します。

> **Note**
>
> スキーマを更新すると、AWS SCT は必要に応じてメタデータをロードします。データベースのスキーマをすべてロードするには、スキーマのコンテキスト (右クリック) メニューを開き、[Load schema] (スキーマの読み込み) を選択します。例えば、このオプションを使用して、データベースのメタデータを一度にロードし、オフラインで作業できます。

AWS SCT で変換されたスキーマの保存および適用

AWS Schema Conversion Tool が変換されたスキーマを生成する場合 (AWS SCT を使用してスキーマを変換する で示すように)、変換されたスキーマはターゲット DB インスタンスにはすぐに適用されません。代わりに、変換されたスキーマは、ターゲット DB インスタンスに適用する準備ができるまで、プロジェクトにローカルで保存されます。この機能を使用すると、ターゲット DB エンジンに自動的に変換できないスキーマ項目を使用できます。自動的に変換できない項目の詳細については、AWS SCT を使用した移行評価レポートの作成 を参照してください。

オプションとして、ターゲット DB インスタンスにスキーマを適用する前に、変換されたスキーマをツールで SQL スクリプトとしてファイルに保存することができます。さらにツールで、変換したスキーマをターゲット DB インスタンスに直接適用することもできます。

変換されたスキーマのファイルへの保存

変換されたスキーマを SQL スクリプトとしてテキストファイルに保存できます。これにより、ツールが自動的に変換できない項目に対応するために、AWS SCT から生成された SQL スクリプトを変更できます。ターゲットデータベースに変換されたスキーマを適用するために、ターゲット DB インスタンスで更新されたスクリプトを実行できます。

変換されたスキーマを SQL スクリプトとして保存するには

1. スキーマを選択して、コンテキスト (右クリック) メニューを開きます。
2. [Save as SQL] (SQL として保存) を選択します。
3. ファイルの名前を入力し、[Save] (保存) を選択します。
4. 変換されたスキーマは、次のいずれかのオプションを使用して保存できます。
   - [単一ファイル]
   - [Single file per stage] (ステージごとに 1 つのファイル)
   - [Single file per statement] (ステートメントごとに 1 つのファイル)

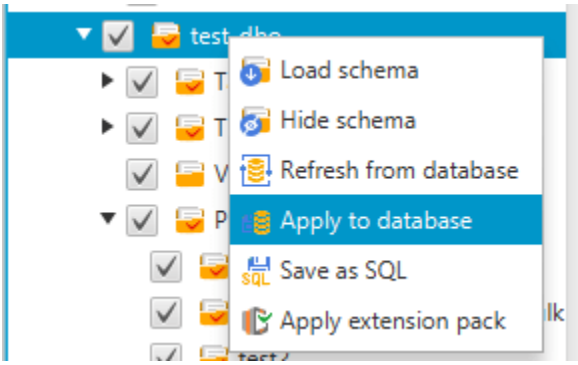
SQL スクリプトの形式を選択するには

1. [Settings] (設定) メニューから [Project settings] (プロジェクト設定) を選択します。
2. [Save scripts] (スクリプトを保存) を選択します。
3. [Vendor] (ベンダー) で、データベースプラットフォームを選択します。

4. [Save SQL scripts to] (SQL スクリプトの保存先) で、データベーススキーマスクリプトの保存方法を選択します。
5. [OK] を選択して、設定を適用します。

変換されたスキーマの適用

変換されたスキーマをターゲット Amazon RDS DB インスタンスに適用する準備ができたら、プロジェクトの右側のパネルからスキーマ要素を選択します。スキーマ要素のコンテキスト (右クリック) メニューを開き、次に示すように、[Apply to database] (データベースに適用) を選択します。



拡張パックスキーマ

変換後のスキーマをターゲット DB インスタンスに適用すると、AWS SCT によって追加ワークスキーマがターゲット DB インスタンスに追加されます。このスキーマは、変換されたスキーマをターゲット DB インスタンスに書き込むときに必要なソースデータベースのシステム関数を実装します。追加されたスキーマは、拡張パックスキーマと呼ばれます。

この拡張パックスキーマは変更しないでください。変更すると、ターゲット DB インスタンスに書き込まれる変換されたスキーマに予期しない結果が発生する可能性があります。スキーマがターゲット DB インスタンスに完全に移行され、AWS SCT が必要なくなった場合は、拡張パックスキーマを削除できます。

拡張パックスキーマの名前は、ソースデータベースに従って次のように記述されます。

- IBM Db2 LUW: `aws_db2_ext`
- Microsoft SQL Server: `aws_sqlserver_ext`
- MySQL: `aws_mysql_ext`
- Oracle: `aws_oracle_ext`
- PostgreSQL: `aws_postgresql_ext`

- SAP ASE: `aws_sapase_ext`

詳細については、「AWS LambdaAWS SCT エクステンションパックの関数を使用する 」を参照してください。

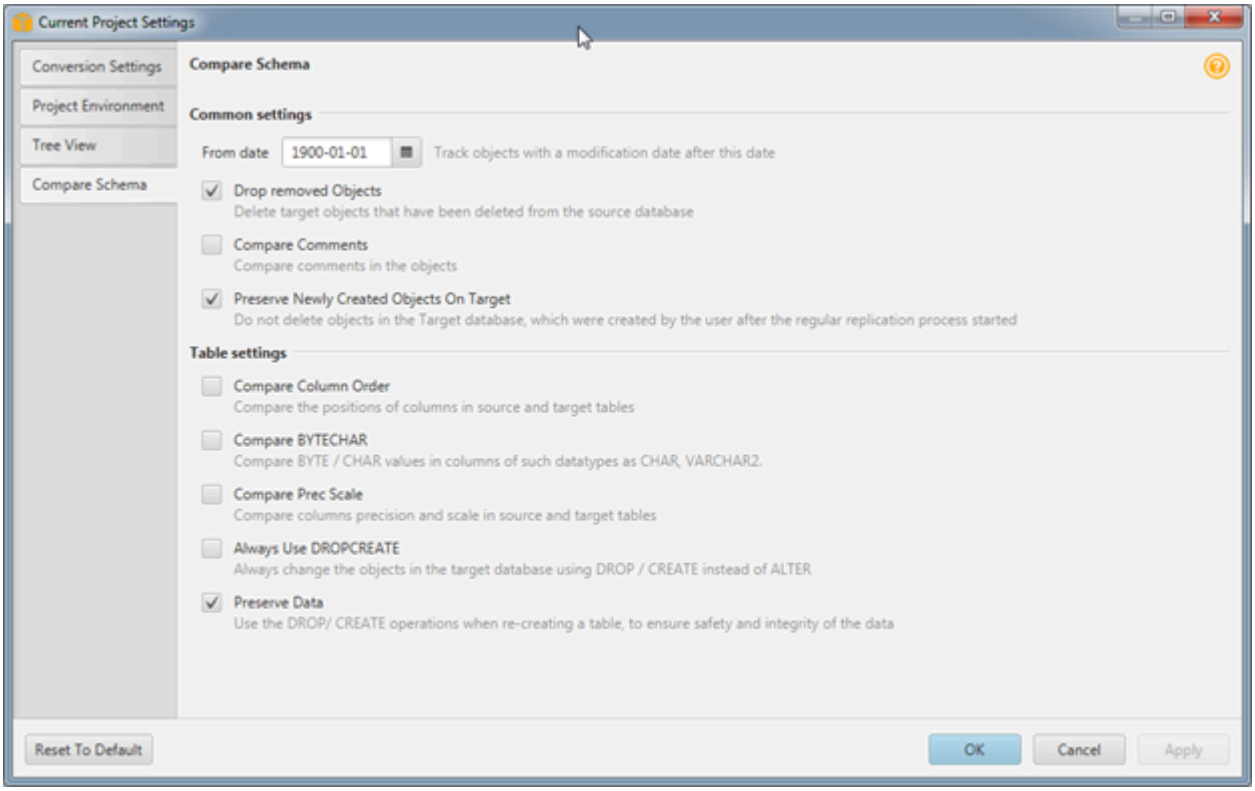
データベーススキーマの比較

移行後にソースまたはターゲットスキーマを変更した場合、AWS SCT を使用して 2 つのデータベーススキーマを比較できます。ソーススキーマと同じか、それよりも前のバージョンのスキーマを比較できます。

次のスキーマ比較がサポートされています。

- Oracle から Oracle、バージョン 12.1.0.2.0、11.1.0.7.0、11.2.0.1.0、10
- SQL Server から SQL Server、バージョン 2016、2014、2012、2008 RD2、2008
- PostgreSQL から PostgreSQL および Aurora PostgreSQL 互換エディション　バージョン 9.6、9.5.9、9.5.4
- MySQL から MySQL、バージョン 5.6.36、5.7.17、5.5

スキーマ比較の設定は、[Project Settings] (プロジェクト設定) ページの [Compare Schema] (スキーマの比較) タブで行います。

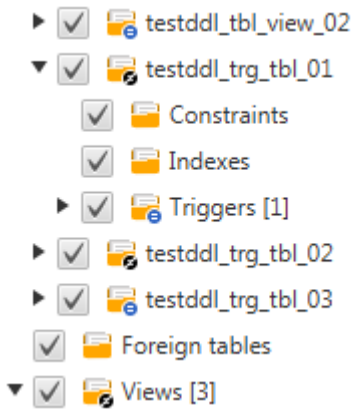


スキーマを比較するには、スキーマを選択します。AWS SCT は、2 つのスキーマ間で異なるオブジェクトと、そうでないオブジェクトを示します。

2 つのスキーマを比較するには

1. 既存の AWS SCT プロジェクトを開くか、プロジェクトを作成し、ソースとターゲットエンドポイントに接続します。
2. 比較するスキーマを選択します。
3. コンテキストメニューを開き (右クリック)、[Compare Schema] (スキーマの比較) を選択します。

AWS SCT は、オブジェクトのアイコンに黒い円を追加して、2 つのスキーマ間で異なるオブジェクトを示します。



スキーマの比較結果は、単一のオブジェクト、オブジェクトの単一のカテゴリ、またはスキーマ全体に適用できます。結果を適用するカテゴリ、オブジェクト、またはスキーマの横にあるボックスをオンにします。

関連する変換オブジェクトの検索

スキーマの変換後に、場合によっては、AWS SCT によってソースデータベースの 1 つのスキーマオブジェクトに対して複数のオブジェクトが作成されることがあります。例えば、Oracle から PostgreSQL への変換を実行するときに、AWS SCT は各 Oracle トリガーを受け取り、それを PostgreSQL ターゲットのトリガーおよびトリガー関数に変換します。また、AWS SCT が Oracle パッケージ関数やプロシージャを PostgreSQL に変換するときに、そのプロシージャまたは関数を実行する前に初期ブロックとして実行する同等の関数と INIT 関数を作成します。

次の手順では、スキーマ変換後に作成された関連するすべてのオブジェクトを表示できます。

スキーマ変換中に作成された関連するオブジェクトを表示するには

1. スキーマ変換後に、ターゲットツリービューで変換されたオブジェクトを選択します。
2. [Related Converted Objects] (関連する変換オブジェクト) タブを選択します。
3. 関連するターゲットオブジェクトのリストを表示します。

AWS SCT を使用したデータウェアハウススキーマの Amazon Redshift への変換

AWS Schema Conversion Tool (AWS SCT) はデータウェアハウススキーマを Amazon Redshift データベーススキーマに変換する多くのプロセスを自動化します。ソースおよびターゲットデータベースエンジンは様々な特徴と機能を備えている可能性があるため、AWS SCT はターゲットデータベースインスタンスにおいて可能な限り同等のスキーマを作成するよう試みます。直接変換ができない場合、AWS SCT はユーザーに対して可能なアクションの一覧を含む評価レポートを提供します。AWS SCT を使用して、キーの管理、データ型とオブジェクトのマッピング、手動変換の作成を行うことができます。

AWS SCT では次のデータウェアハウススキーマを Amazon Redshift に変換できます。

- Amazon Redshift
- Azure Synapse Analytics (バージョン 10)
- BigQuery
- Greenplum データベース (バージョン 4.3)
- Microsoft SQL Server (バージョン 2008 以降)
- Netezza (バージョン 7.0.3 以降)
- Oracle (バージョン 10.2 以降)
- Snowflake (バージョン 3)
- Teradata (バージョン 13 以降)
- Vertica (バージョン 7.2 以降)

オンライン トランザクション処理 (OLTP) データベース スキーマの変換については、「AWS SCT を使用したデータベーススキーマの変換」を参照してください。

データウェアハウススキーマを変換するには、次のステップを実行します。

1. 最適化戦略とルールを指定し、AWS SCT で使用する移行ルールを指定します。列のデータ型の変更、あるスキーマから別のスキーマへのオブジェクトの移動、オブジェクトの名前の変更を行うルールを設定できます。

最適化と移行ルールは、[Settings] (設定) で指定できます。最適化方法の詳細については、「AWS SCT で使用する最適化戦略とルールの選択」を参照してください。移行ルールの詳細については、「AWS SCT での移行ルールの作成」を参照してください。

2. ソースデータウェアハウスの統計を提供し、AWS SCT がデータウェアハウスの変換方法を最適化できるようにします。データベースから統計を直接収集するか、既存の統計ファイルをアップロードできます。データウェアハウスの統計の提供については、「AWS SCT の統計の収集またはアップロード」を参照してください。
3. 自動的に変換できないスキーマの要素の詳細を記載したデータベース移行評価レポートを作成します。ソースデータベースと互換性のあるターゲットデータベースで、スキーマを手動で作成する必要があるかどうかを識別するため、このレポートを使用できます。評価レポートの詳細については、「AWS SCT を使用した移行評価レポートの作成」を参照してください。
4. スキーマの変換。AWS SCT は確認のために変換されたスキーマのローカルバージョンを作成しますが、準備ができるまでターゲットデータベースには適用されません。変換の詳細については、「AWS SCT を使用したスキーマの変換」を参照してください。
5. スキーマの変換後は、キーを管理し、編集できます。キー管理はデータウェアハウス変換の柱となります。キー管理の詳細については、「AWS SCT のキーの管理とカスタマイズ」を参照してください。
6. 自動的に変換できないスキーマの要素がある場合、2 つの選択肢があります。ソーススキーマを更新したのち再び変換するか、またはターゲットデータベースにおいて同等のスキーマの要素を作成します。スキーマ要素の手動変換の詳細については、「AWS SCT でのマニュアル変換の処理」を参照してください。ソーススキーマの更新の詳細については、「AWS SCT で変換されたスキーマの更新および再読み込み」を参照してください。
7. 準備ができたら、変換されたスキーマをターゲットデータベースに適用できます。変換されたスキーマの保存と適用の詳細については、「AWS SCT で変換されたスキーマの保存および適用」を参照してください。

ターゲットとしての Amazon Redshift の許可

次のリストは、Amazon Redshift をターゲットとして使用するためのアクセス許可を以下に示します。

- CREATE ON DATABASE — データベースに新しいスキーマを作成できます。
- CREATE ON SCHEMA — データベーススキーマにオブジェクトを作成できます。
- GRANT USAGE ON LANGUAGE — データベースに新しい関数やプロシージャを作成できます。

- スキーマ pg_catalog のすべてのテーブルに対して選択権限を付与 — Amazon Redshift クラスターに関するシステム情報をユーザーに提供します。
- pg_class_info での GRANT SELECT — テーブル分散スタイルに関する情報をユーザーに提供します。

次のコード例を使用してデータベースユーザーを作成し、権限を付与できます。

```
CREATE USER user_name PASSWORD your_password;
GRANT CREATE ON DATABASE db_name TO user_name;
GRANT CREATE ON SCHEMA schema_name TO user_name;
GRANT USAGE ON LANGUAGE plpythonu TO user_name;
GRANT USAGE ON LANGUAGE plpgsql TO user_name;
GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog TO user_name;
GRANT SELECT ON pg_class_info TO user_name;
GRANT SELECT ON sys_serverless_usage TO user_name;
GRANT SELECT ON pg_database_info TO user_name;
GRANT SELECT ON pg_statistic TO user_name;
```

前の例では、*user_name* を自分のユーザーの名前に置き換えてください。次に、*db_name* をターゲットの Amazon Redshift データベースの名前に置き換えます。次に、*schema_name* を Amazon Redshift スキーマの名前に置き換えます。変換したコードを適用するか、データを移行するターゲットスキーマごとに、GRANT CREATE ON SCHEMA の操作を繰り返します。最後に、*your_password*を安全なパスワードに置き換えます。

ターゲットの Amazon Redshift データベースに拡張パックを適用できます。拡張パックは、オブジェクトを Amazon Redshift に変換するときに必要なソース データベース機能をエミュレートするアドオンモジュールです。詳細については、「AWS SCT 拡張パックの使用」を参照してください。

このオペレーションでは、AWS SCT ではお客様に代わって Amazon S3 バケットへのアクセス許可が必要となります。このアクセス許可を提供するには、次のポリシーを持つ AWS Identity and Access Management (IAM) ユーザーを作成します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:DeleteObject",
            "s3:GetObject",
```

```
            "s3:ListBucket",
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-sct-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":""
      }
   ]
}
```

AWS SCT で使用する最適化戦略とルールの選択

AWS Schema Conversion Tool がデータウェアハウススキーマを変換する方法を最適化するため、ツールで使用する戦略とルールを選択できます。スキーマを変換し、推奨されたキーを確認した後で、目的の結果を得るためにルールを調整するか、戦略を変更することができます。

最適化戦略とルールを選択するには

1. [Settings] (設定) を選択し、[Project Settings] (プロジェクト設定) を選択します。[Current project settings] (現在のプロジェクト設定) ダイアログボックスが表示されます。
2. 左側のペインで、[Optimization Strategies] (最適化戦略) を選択します。最適化戦略が、デフォルト値が選択されて右側のペインに表示されます。
3. [Strategy Sector] (戦略セクター) で、使用する最適化戦略を選択します。次から選択できます。

   - [Use metadata, ignore statistical information] (メタデータを使用して、統計情報は無視する) – この戦略では、メタデータからの情報のみが、最適化の決定に使用されます。例えば、ソーステーブルに複数のインデックスがある場合、ソースデータベースのソート順が使用され、最初のインデックスが分散キーになります。

   - [Ignore metadata, use statistical information] (メタデータを無視して、統計情報を使用する) – この戦略では、統計情報のみが、最適化の決定に使用されます。この戦略は、統計が提供され

るテーブルと列のみに適用されます。詳細については、「AWS SCT の統計の収集またはアップロード」を参照してください。

- [Use metadata and use statistical information] (メタデータおよび統計情報を使用する) – この戦略では、最適化の決定にメタデータおよび統計の両方が使用されます。

4. 最適化戦略を選択した後で、使用するルールを選択できます。次から選択できます。

   - メタデータを使用して分散キーとソートキーを選択
   - 照合用のファクトテーブルと適切なディメンションを選択
   - インデックスの列の基数を分析
   - クエリログテーブルから最も使用されるテーブルと列を見つける

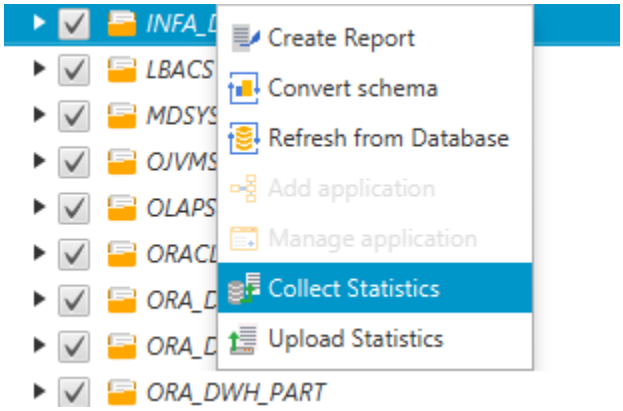
   ルールごとに、ソートキーの重みと分散キーの重みを入力できます。AWS SCT では、スキーマの変換時に、ここで選択した重みが使用されます。後で推奨されたキーを確認するときに、結果に満足できない場合は、ここに戻って設定を変更できます。詳細については、「AWS SCT のキーの管理とカスタマイズ」を参照してください。

AWS SCT の統計の収集またはアップロード

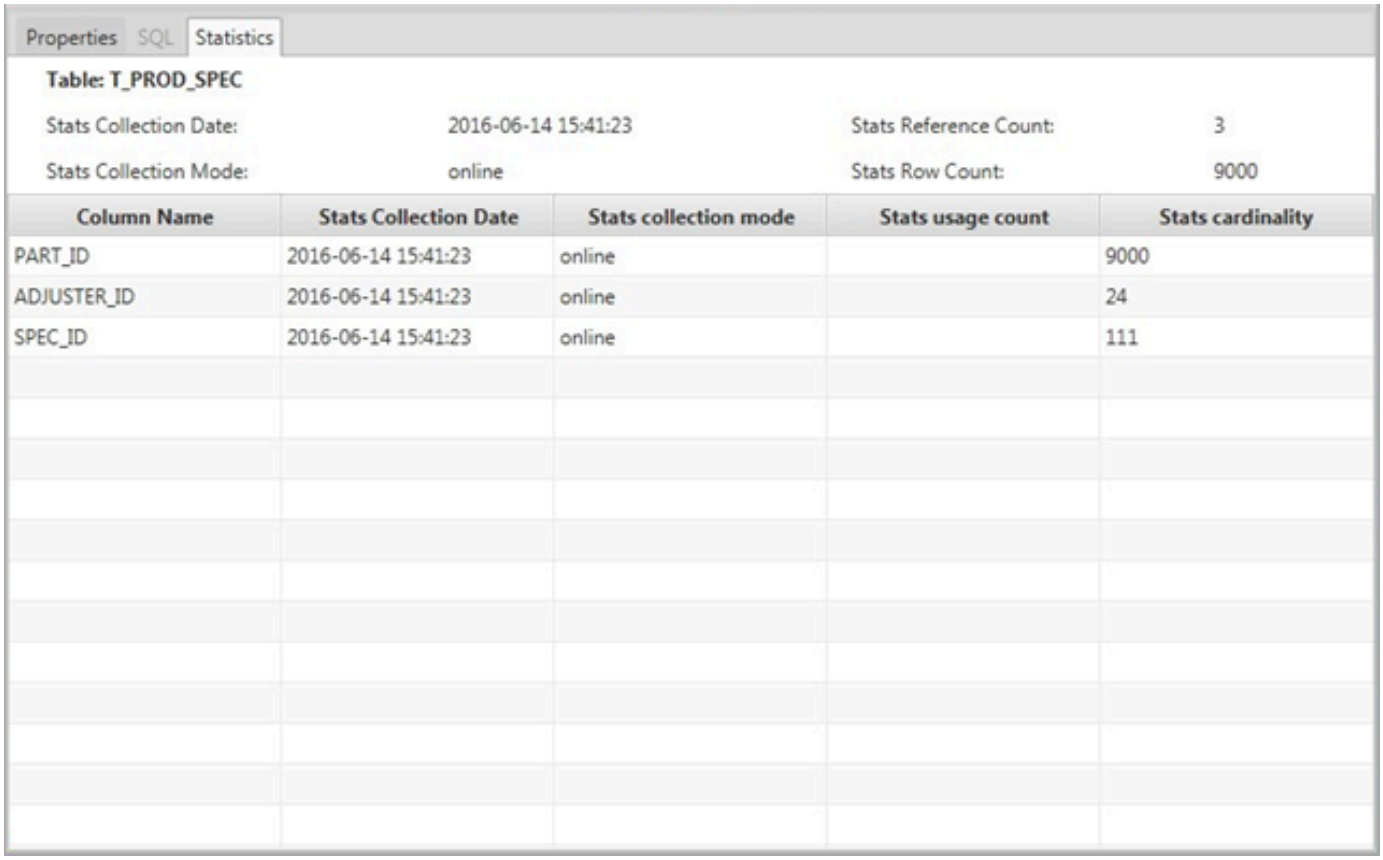
AWS Schema Conversion Tool がデータウェアハススキーマを変換する方法を最適化するには、ツールが使用できるソースデータベースから統計を提供できます。データベースから統計を直接収集するか、既存の統計ファイルをアップロードできます。

統計を提供し、確認するには

1. プロジェクトを開き、ソースデータベースに接続します。
2. プロジェクトの左側のパネルからスキーマオブジェクトを選択し、オブジェクトのコンテキスト (右クリック) メニューを開きます。次に示すように、[Collect Statistics] (統計の収集) または [Upload Statistics] (統計のアップロード) を選択します。



3. プロジェクトの左側のパネルからスキーマオブジェクトを選択し、[Statistics] (統計) タブを選択します。オブジェクトの統計を確認できます。



後で推奨されたキーを確認するときに、結果に満足できない場合は、追加の統計を収集して、この手順を繰り返すことができます。詳細については、「AWS SCT のキーの管理とカスタマイズ」を参照してください。

AWS SCT での移行ルールの作成

AWS SCT でスキーマを変換する前に、ルールを設定できます。列のデータ型の変更、あるスキーマから別のスキーマへのオブジェクトの移動、オブジェクト名の変更などを、移行ルールを設定して実行できます。例えば、test_TABLE_NAME という名前のソーススキーマにテーブルがあるとします。ターゲットスキーマ内にある プレフィックス test_ をプレフィックス demo_ に変更するルールを設定できます。

> **Note**
>
> 移行ルールは、異なるソースとターゲットのデータベースエンジンに対してのみ作成できます。

移行ルールを作成して実行できるタスクは次のとおりです。

- プレフィックスの追加、削除、または置換
- サフィックスの追加、削除、または置換
- 列照合の変更
- データタイプの変更
- char、varchar、nvarchar、string およびデータ型の長さを変更
- オブジェクトの移動
- オブジェクトの名前変更

移行ルールを作成できるオブジェクトは次のとおりです。

- データベース
- スキーマ
- テーブル
- Column

移行ルールの作成

移行ルールを作成し、プロジェクトの一部として保存できます。プロジェクトを開いた状態で、以下の手順で移行ルールを作成します。

移行ルールを作成するには

1. [View] (ビュー) メニューで、[Mapping View] (マッピングビュー) を選択します。
2. [Server mappings] (サーバーマッピング) で、ソースサーバとターゲットサーバのペアを選択します。
3. [New migration rule] (新しい移行ルール) を選択します。[Transformation rules] (変換ルール) ダイアログボックスが表示されます。
4. [Add new rule] (新しいルールを追加) を選択します。ルールのリストに新しい行が追加されます。
5. ルールを設定します。

   a. [Name] (名前) に、ルールの名前を入力します。

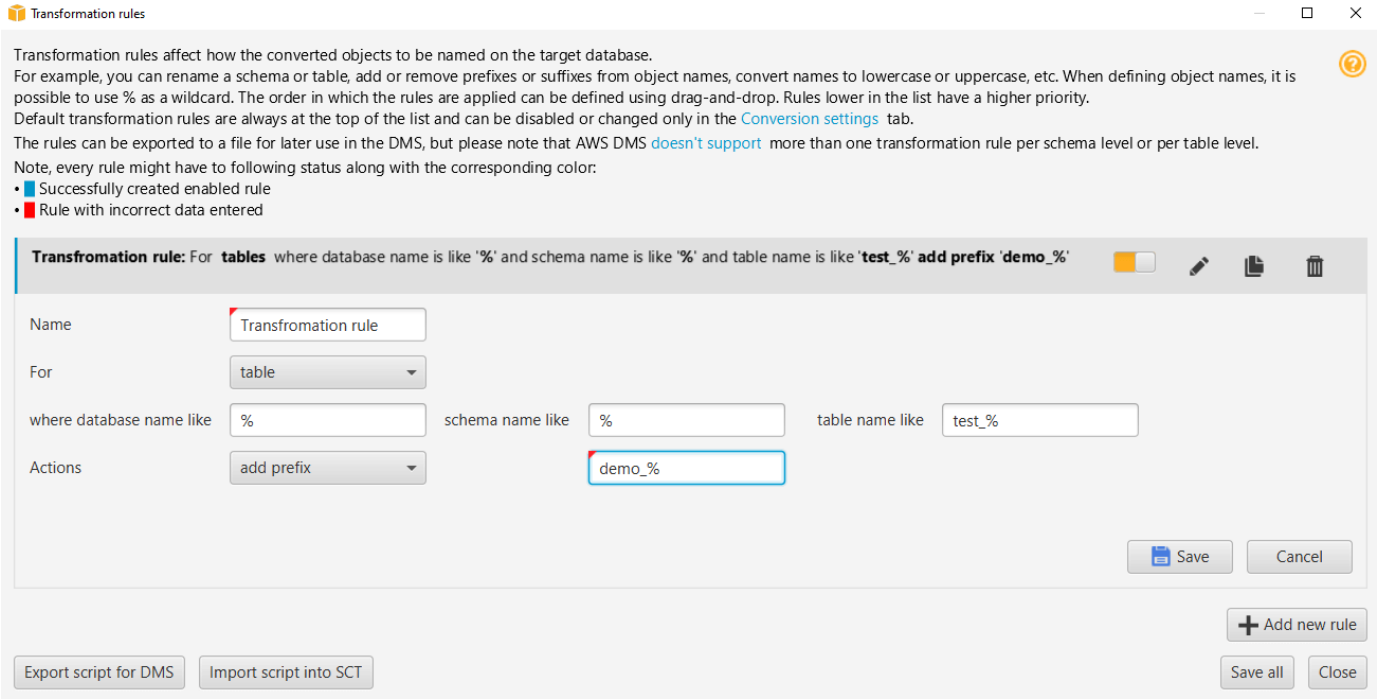
   b. [For] で、ルールを適用するオブジェクトのタイプを選択します。

   c. [where] (条件) に、オブジェクトに適用するフィルターを入力後、移行ルールを適用します。where 句は、like 句を使用して評価されます。正確な名前を入力して特定のオブジェクトを選択するか、パターンを入力して複数のオブジェクトを選択できます。

      where 句で利用可能なフィールドは、オブジェクト型によって異なります。例えば、スキーマの名前のオブジェクト型がスキーマの場合、使用可能なフィールドは 1 つだけです。

   d. [Actions] (アクション) で、作成する移行ルールを選択します。

   e. ルールタイプに応じて、追加の値を 1 つまたは 2 つ入力します。例えば、オブジェクトの名前を変更するには、オブジェクトの新しい名前を入力します。プレフィックスを置換するには、現在のプレフィックスおよび置換後のプレフィックスを入力します。
6. 移行ルールを設定したら、[Save] (保存) を選択してルールを保存します。変更をキャンセルする場合は、[Cancel] (キャンセル) を選択します。



7. ルールの追加、編集、削除が完了したら、[Save All] (すべてを保存) を選択して変更内容を保存します。
8. [Close] (閉じる) を選択して、[Transformation rules] (変換ルール) ダイアログボックスを閉じます。

移行ルールを削除せずに無効にするには、切り替えアイコンを使用できます。既存の移行ルールを複製するには、コピーアイコンを使用します。既存の移行ルールを編集するには、鉛筆アイコンを使用します。既存の移行ルールを削除するには、削除アイコンを使用します。移行ルールへの変更を保存するには、[Save All] (すべて保存) を選択します。

移行ルールのエクスポート

AWS Database Migration Service (AWS DMS) を使用してソースデータベースからターゲットデータベースにデータを移行する場合は、移行ルールに関する情報を AWS DMS に入力します。タスクに関する詳細は、「AWS Database Migration Service のレプリケーションタスクの使用」を参照してください。

移行ルールをエクスポートするには

1. AWS Schema Conversion Toolで、[View] (ビュー) メニューで [Mapping View] (マッピングビュー) を選択します。

2. [Migration rules] (移行ルール) で、移行ルールを選択してから、[Modify migration rule] (移行ルールの変更) を選択します。
3. [AWS DMS のスクリプトのエクスポート] を選択します。
4. スクリプトを保存する場所を指定し、[Save] (保存) を選択します。移行ルールは、AWS DMS で使用できる JSON スクリプトとして保存されます。

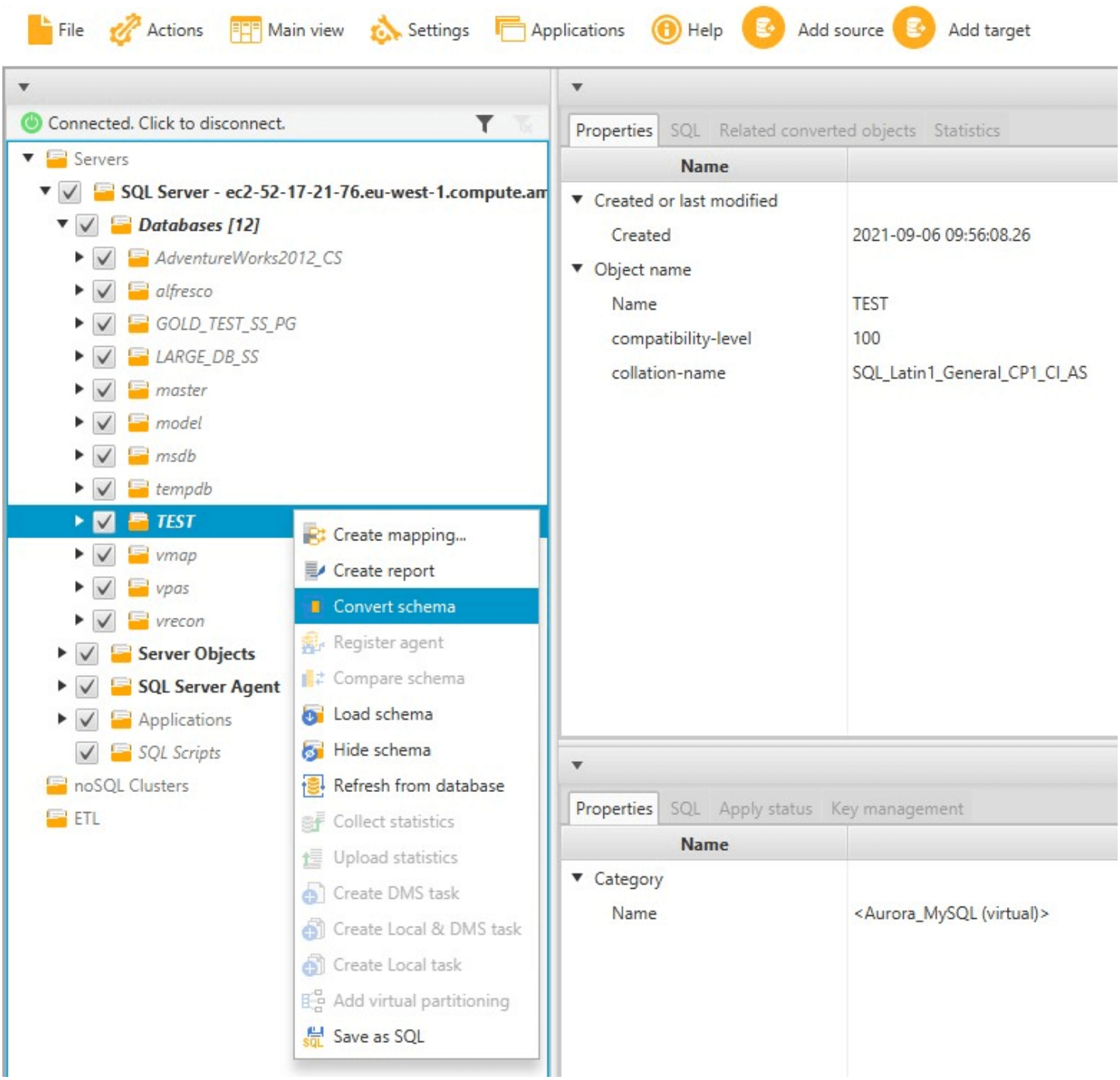
AWS SCT を使用したスキーマの変換

ソースデータベースおよびターゲットデータベースの両方にプロジェクトを接続したのち、AWS Schema Conversion Tool プロジェクトは左のパネルでソースデータベースからのスキーマを表示します。スキーマはツリービュー形式で表示され、ツリーの各ノードは遅延ロードです。ツリービューでノードを選択すると、その時点で AWS SCT はソースデータベースからスキーマ情報をリクエストします。

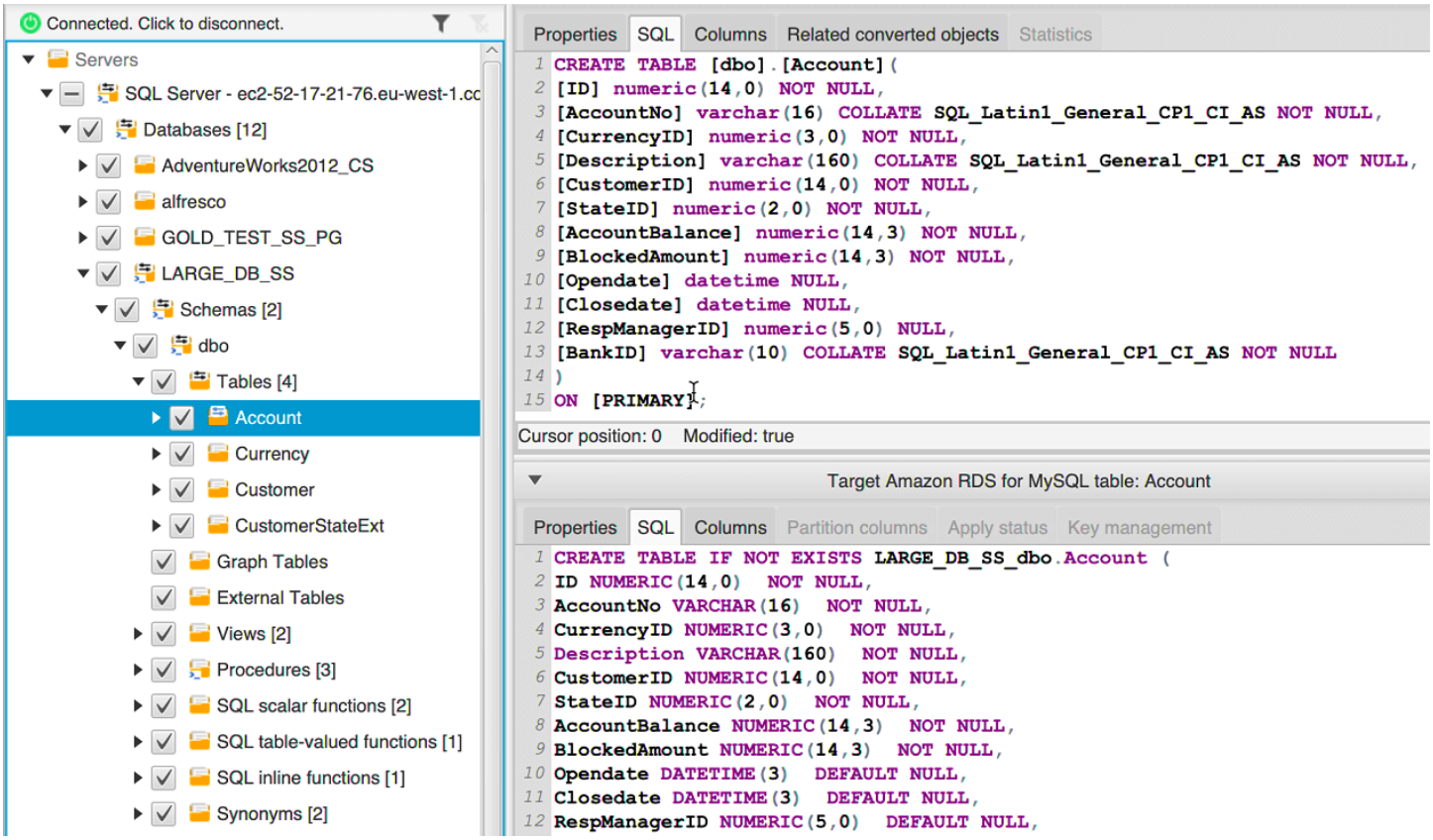
ソースデータベースからスキーマ項目を選択し、ターゲットデータベースのデータベースエンジンと同等のスキーマにスキーマを変換することができます。ソースデータベースから任意のスキーマ項目を選択して変換できます。選択したスキーマ項目が親項目に依存する場合、AWS SCT はその親項目にもスキーマを生成します。例えば、変換する列をテーブルから選択する場合、AWS SCT はスキーマを列、その列のあるテーブル、そのテーブルのあるデータベースに生成します。

スキーマの変換

ソースデータベースからスキーマを変換するには、変換するスキーマの名前のチェックボックスを選択します。次に、プロジェクトの左側のパネルからこのスキーマを選択します。 AWS SCT で、スキーマ名が青で強調表示されます。スキーマのコンテキスト（右クリック）メニューを開き、次に示すように [スキーマを変換] を選択します。



ソースデータベースからスキーマを変換すると、プロジェクトの左のパネルからスキーマ項目を選択でき、プロジェクトの中央のパネルで変換されたスキーマを表示できます。下中央のパネルに変換されたスキーマを作成する SQL コマンドのプロパティが次のように表示されます。



スキーマを変換した後、プロジェクトを保存できます。ソースデータベースからのスキーマ情報は、プロジェクトと共に保存されます。この機能は、ソースデータベースに接続せずにオフラインで作業できることを意味します。ソースデータベースに [Refresh from Database] (データベースから更新) が選択されている場合、AWS SCT は必要に応じて、ソースデータベースに接続してプロジェクトのスキーマを更新します。詳細については、「AWS SCT で変換されたスキーマの更新および再読み込み」を参照してください。

自動的に変換できない項目のデータベース移行評価レポートを作成できます。評価レポートは、自動的に変換できないスキーマ項目の特定と解決に役立ちます。詳細については、「AWS SCT を使用した移行評価レポートの作成」を参照してください。

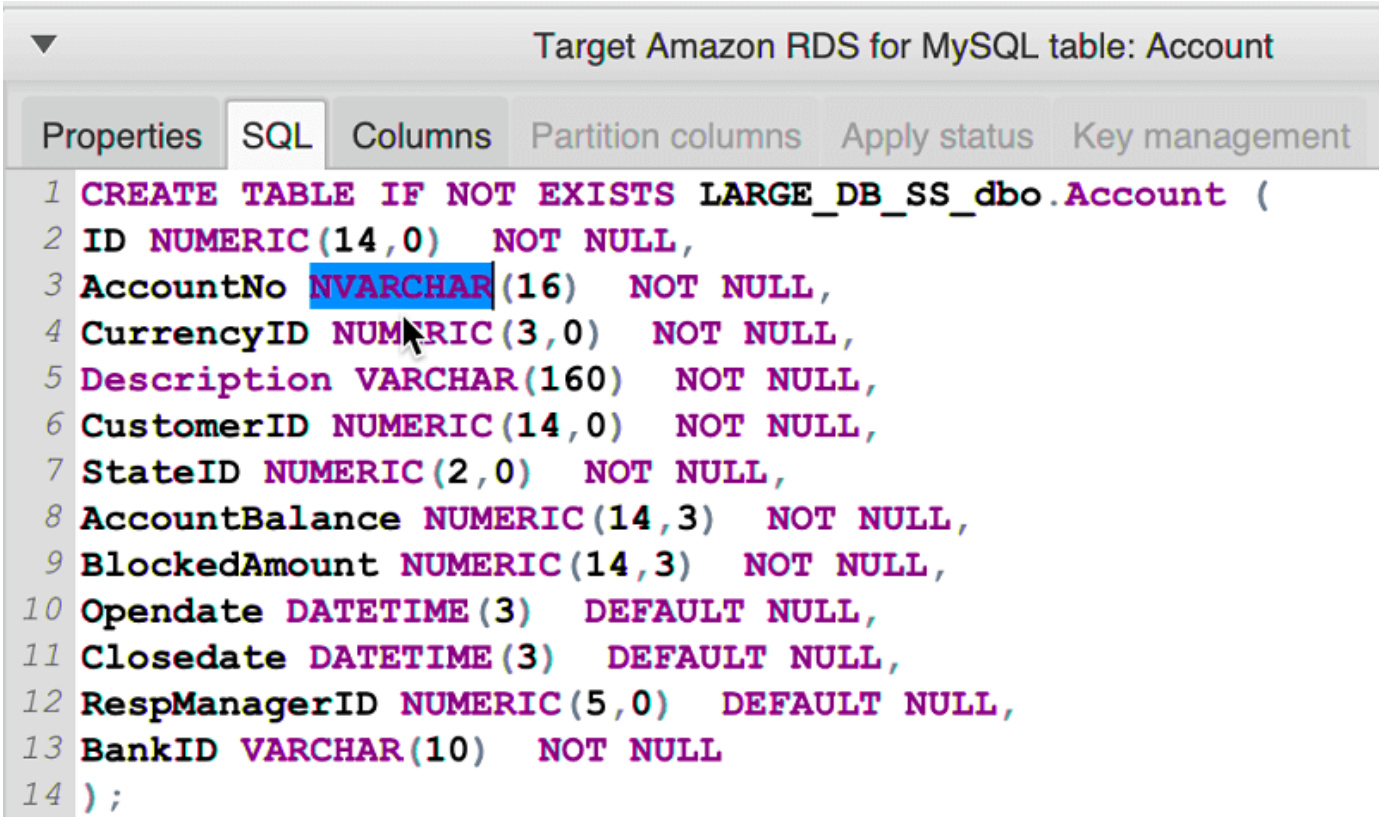
AWS SCT が変換されたスキーマを生成する場合、そのスキーマはターゲットデータベースにすぐには適用されません。代わりに、変換されたスキーマはターゲットデータベースに適用する準備ができるまでローカルに保存されます。詳細については、「変換されたスキーマの適用」を参照してください。

変換されたスキーマの編集

変換されたスキーマを編集し、プロジェクトの一部として変更を保存できます。

変換されたスキーマを編集するには

1. ソースデータベースのスキーマを表示する左のパネルで、変換されたスキーマの編集するスキーマ項目を選択します。
2. 選択した項目の変換されたスキーマを表示している下中央のパネルで [SQL] タブを選択します。
3. [SQL] タブで表示されたテキストで、必要に応じてスキーマを変更します。更新する際に、スキーマはプロジェクトと共に自動的に保存されます。



更新する際に、プロジェクトと共に変換されたスキーマへの変更が格納されます。新しくソースデータベースからスキーマ項目を変換し、項目の以前に変換されたスキーマを更新した場合、既存の更新はソースデータベースに基づいて新しく変換されたスキーマで置き換えられます。

変換されたスキーマのクリア

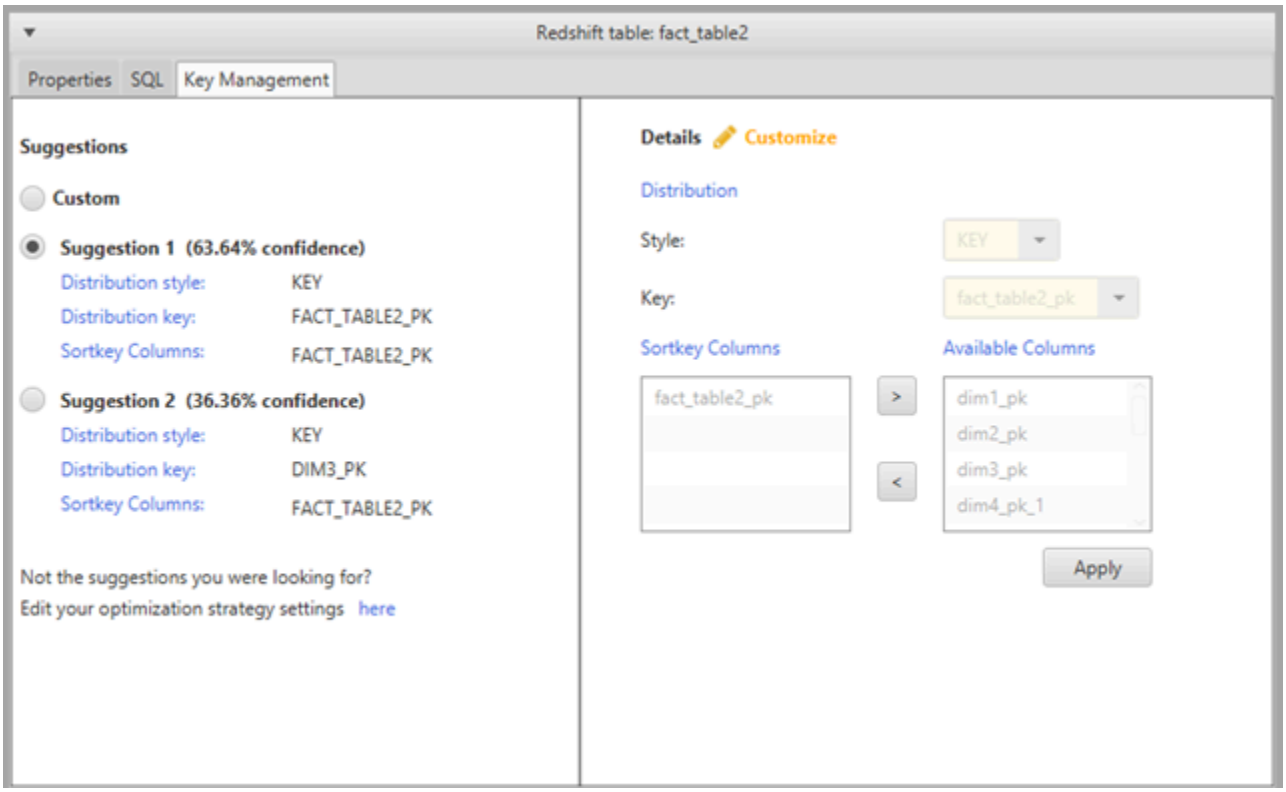
ターゲットデータベースにスキーマを適用するまで、AWS SCT はプロジェクトで変換したスキーマをローカルでのみ保存します。ターゲットデータベース用ツリービューノードを選択して、プロジェクトから計画されたスキーマをクリアし、[Refresh from Database] (データベースから更新) を選択します。ターゲットデータベースにスキーマが書き込まれていなかったため、データベースから更

新することによって、AWS SCT プロジェクトで計画されたスキーマの要素は削除され、ターゲットデータベースに存在するものと一致するようになります。

AWS SCT のキーの管理とカスタマイズ

AWS Schema Conversion Tool を使用してスキーマを変換した後、キーを管理し、編集できます。キー管理はデータウェアハウス変換の柱となります。

キーを管理するには、ターゲットデータベースでテーブルを選択し、次に示すように [Key Management] (キーの管理) タブを選択します。



左側のペインにはキーの提案が含まれ、各提案の信頼性に関する評価が含まれています。いずれかの提案を選択するか、右側のペインで編集してキーをカスタマイズできます。

キーの選択が予期したとおりでない場合、最適化戦略を編集し、変換を再試行できます。詳細については、「AWS SCT で使用する最適化戦略とルールの選択」を参照してください。

関連トピック

- 最良のソートキーの選択
- 最適な分散スタイルの選択

AWS SCT での評価レポートの作成と使用

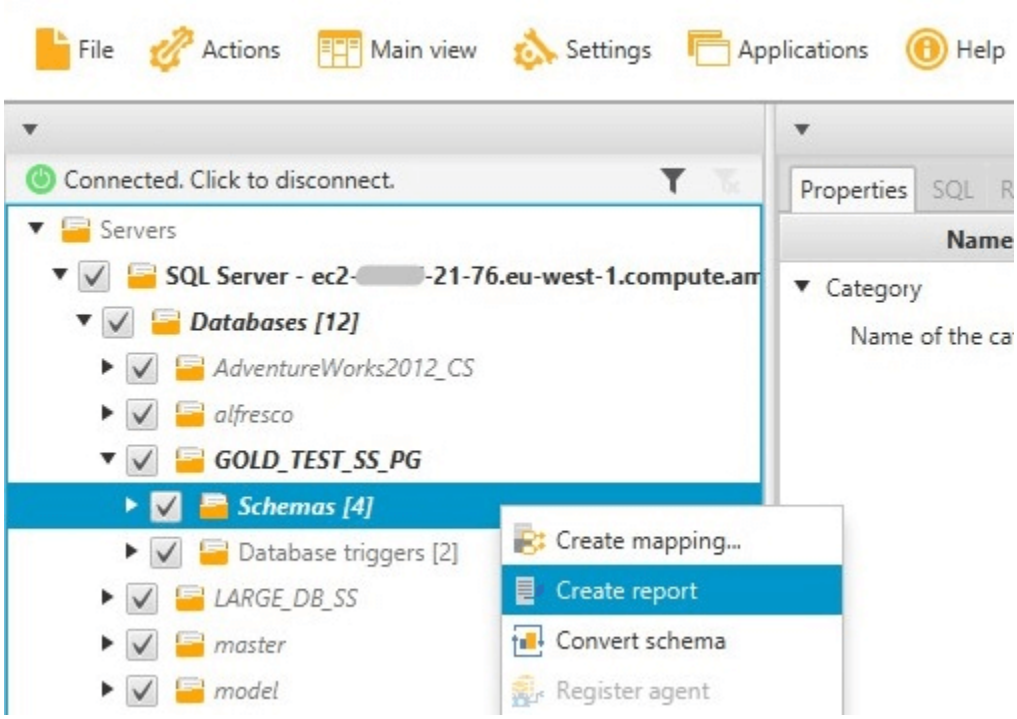
AWS Schema Conversion Tool は、スキーマの変換に役立つようにデータベース移行の評価レポートを作成します。データベース移行評価レポートでは、ソースデータベースからターゲットデータベースへのスキーマの変換に関する重要な情報を得られます。レポートは、ターゲット データベースの DB エンジンに変換できないスキーマに関する、すべてのスキーマ変換タスクやアクション項目の詳細をまとめたものです。このレポートには、自動変換できないターゲットデータベースで同等のコードを記述するために必要な労力の予測も記載されています。

データベース移行評価レポートの作成

次の手順に従ってデータベース移行評価レポートを作成します。

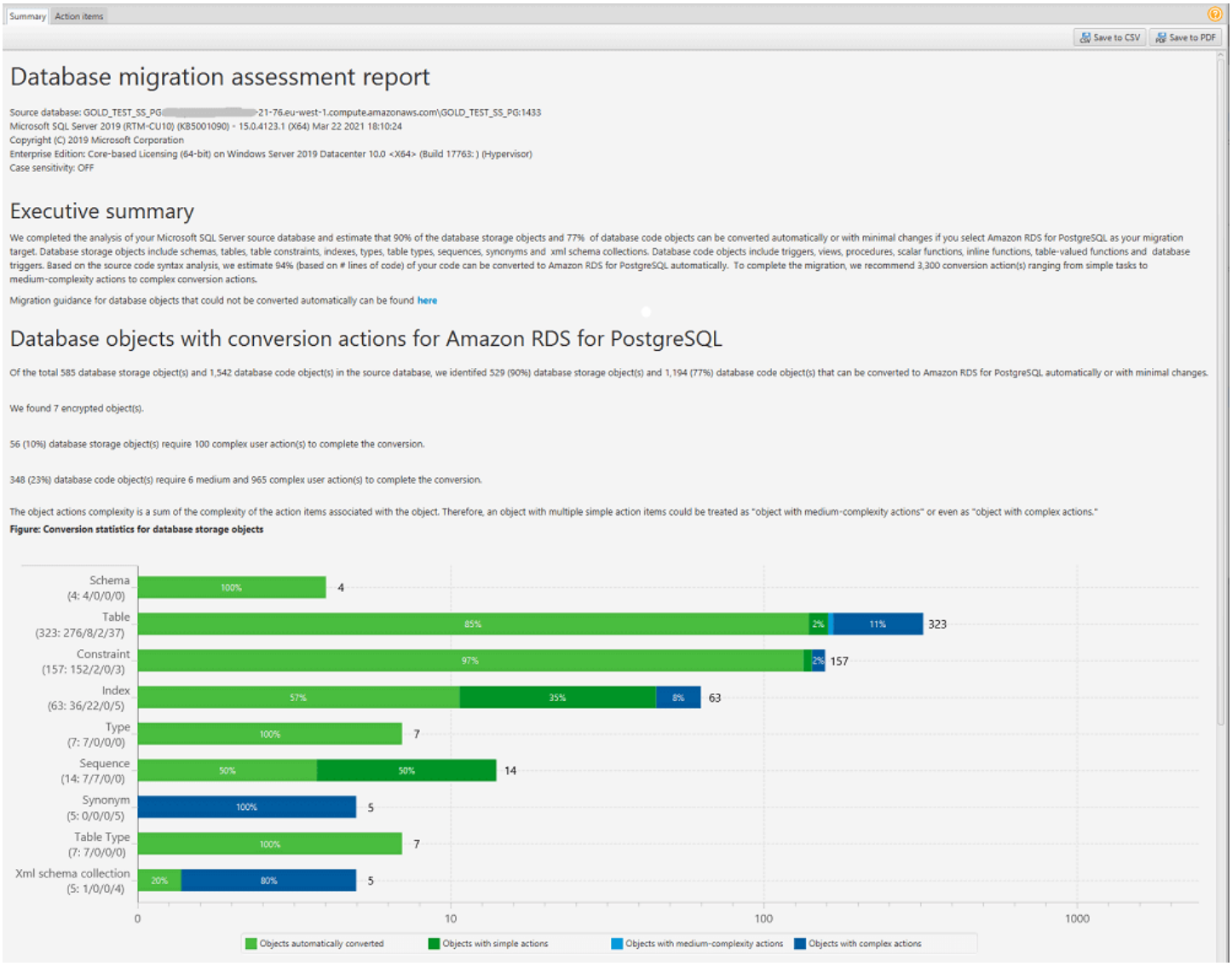
データベース移行評価レポートを作成するには

1. ソースデータベースのスキーマを表示する左のパネルで、評価レポートを作成するスキーマオブジェクトを選択します。
2. オブジェクトのコンテキスト (右クリック) メニューを開き、[Create Report] (レポートを作成) を選択します。



評価レポートの概要

評価レポートを作成すると、評価レポートビューが開き、[Summary] (概要) タブが表示されます。[Summary] (概要) タブには、データベース移行評価レポートの要約が表示されます。自動的に変換された項目と、自動的に変換されなかった項目が表示されます。



ターゲットデータベースエンジンに自動的に変換できないスキーマ項目に関して、要約には、ソースと同等のスキーマ項目をターゲット DB インスタンスで作成するために必要な労力の予測が含まれます。

このレポートは、これらのスキーマアイテムを変換する推定時間を次のように分類します。

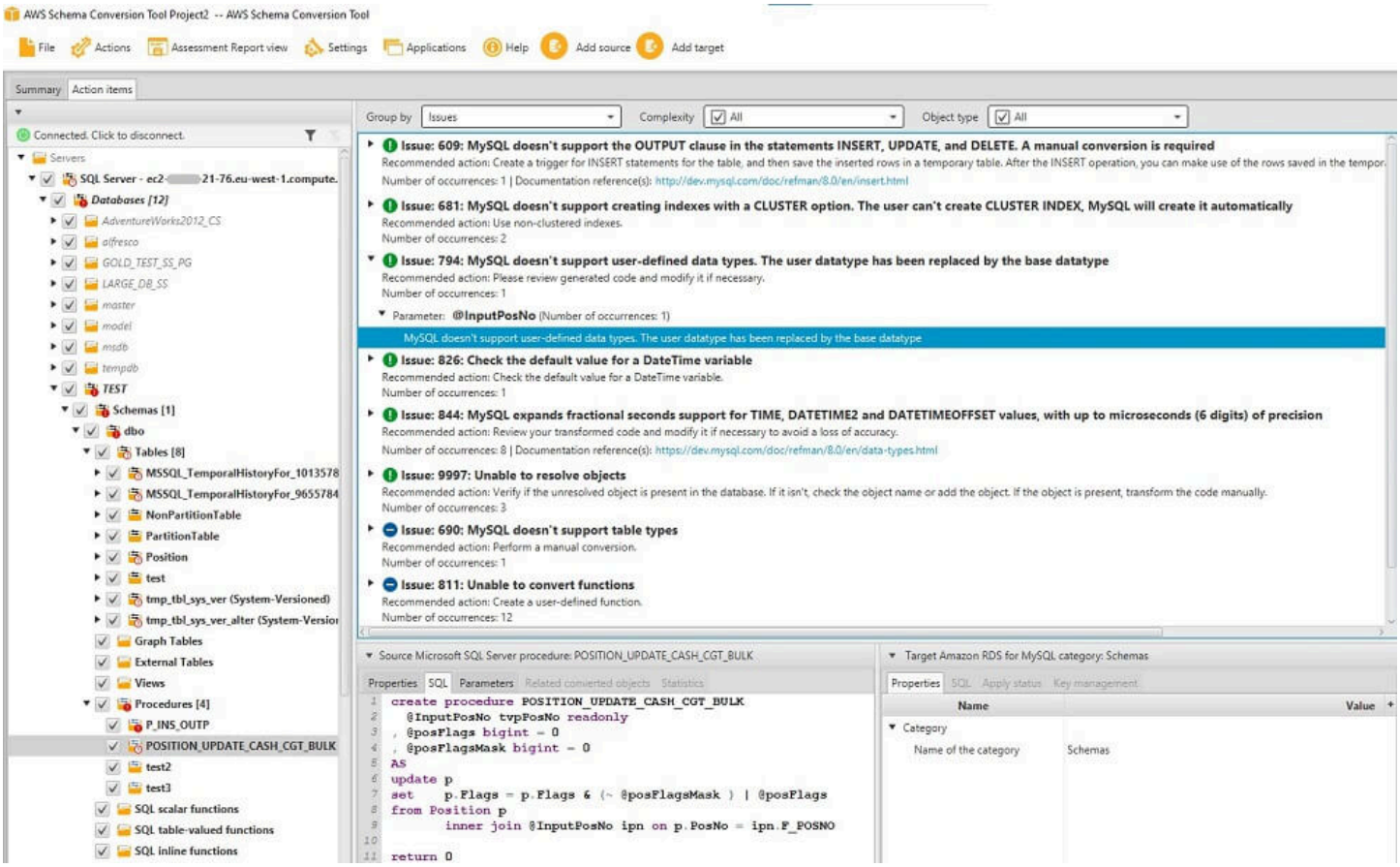
- [Simple] (シンプル) - 1 時間以内に完了できるアクション。
- [Medium] (ミディアム) - より複雑で、1～4 時間で完了できるアクション。

- [Significant] (大規模) - 非常に複雑で、完了に 4 時間以上かかるアクション。

評価レポートアクション項目

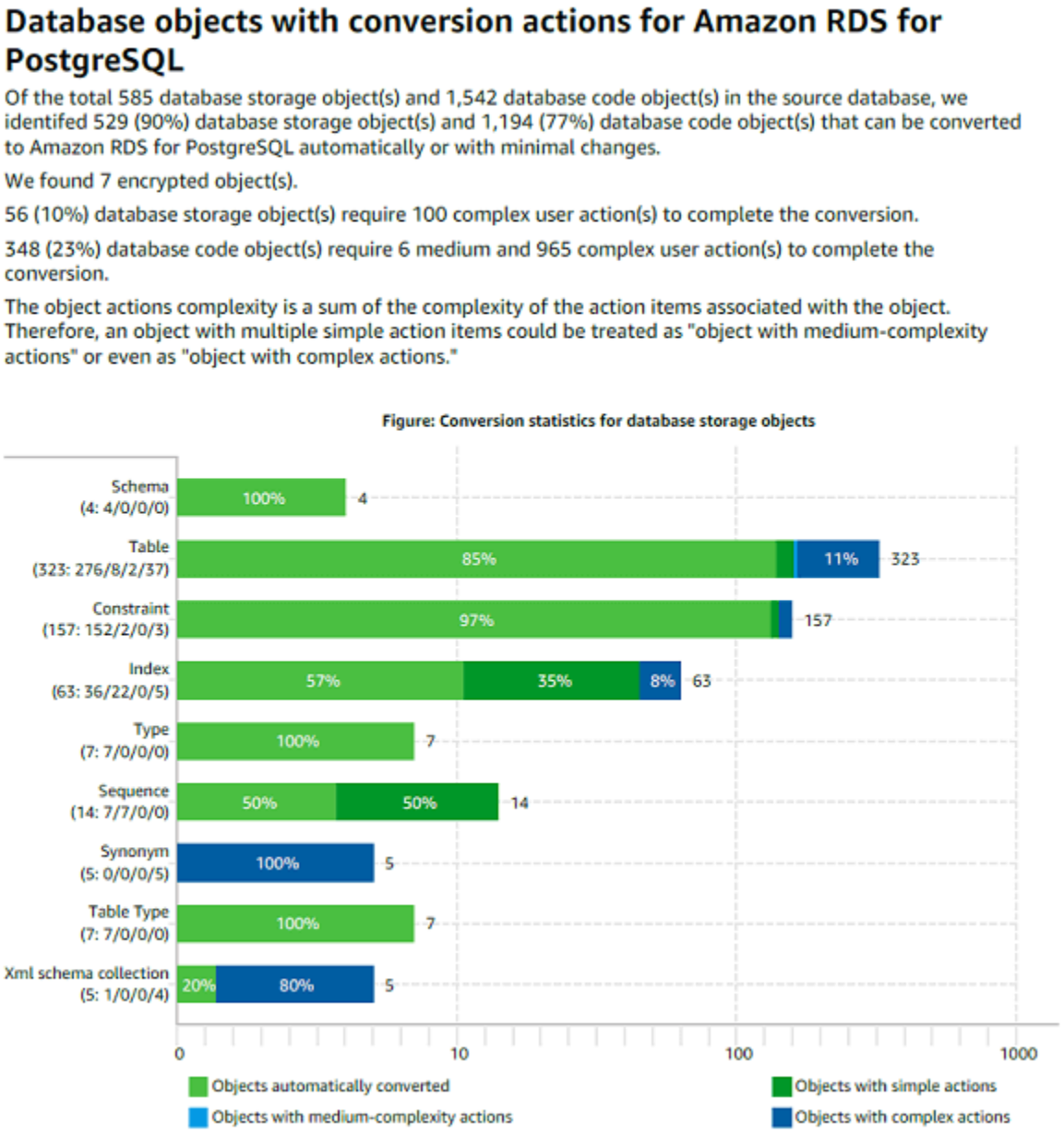
評価レポートビューには [Action Items] (アクション項目) タブも含まれます。このタブには、ターゲットデータベースのデータベースエンジンに自動的に変換できない項目のリストが含まれます。リストからアクション項目を選択する場合、アクション項目が適用されるスキーマが AWS SCT によってハイライトされます。

レポートには、手動でスキーマ項目を変換する方法の推奨事項も含まれています。手動変換の処理方法の決定に関する詳細については、AWS SCT でのマニュアル変換の処理 を参照してください。



評価レポートの保存

データベース移行評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらでも保存できます。CSV ファイルには、アクション項目の情報のみが含まれます。PDF ファイルには、次の例に示すように、要約とアクション項目の情報の両方が含まれます。



AWS SCT でのマニュアル変換の処理

評価レポートには、ターゲットデータベースのデータベースエンジンに自動的に変換できない項目のリストが含まれます。変換できない各項目に対して、[Action Items] (アクション項目) タブにアクション項目があります。

評価レポートのアクション項目には、以下の方法で対応できます。

- ソースデータベーススキーマの変更

- ターゲットデータベーススキーマの変更

ソーススキーマの変更

一部の項目については、ソースデータベースのデータベーススキーマを、自動的に変換できるスキーマに変更する方が容易な場合があります。最初に、新しい変更にアプリケーションアーキテクチャと互換性があることを確認し、次にソースデータベースのスキーマを更新します。最後に、更新されたスキーマ情報のあるプロジェクトを更新します。その後、更新されたスキーマを変換し、新しいデータベース移行評価レポートを生成できます。ソーススキーマで変更された項目のアクション項目は表示されなくなります。

このプロセスの利点は、ソースデータベースから更新するときに、更新されたスキーマを常に使用できることです。

ターゲットスキーマの変更

一部の項目については、ターゲットデータベースに変換されたスキーマを適用する方が容易な場合があるため、自動的に変換できなかった項目のターゲットデータベースに同等のスキーマを手動で追加します。スキーマを適用することで、ターゲットデータベースに自動的に変換できるスキーマすべてを記述できます。詳細については、「AWS SCT で変換されたスキーマの保存および適用」を参照してください。

ターゲットデータベースに書き込まれるスキーマには、自動的に変換できなかった項目は含まれません。ターゲットデータベースにスキーマを適用したら、ターゲットデータベースでソースデータベースと同等のスキーマを手動で作成できます。データベース移行評価レポートのアクション項目には、同等のスキーマを作成する方法の提案が含まれています。

> **Warning**
>
> ターゲットデータベースでスキーマを手動で作成する場合、手動作業のコピーを保存します。再度プロジェクトからターゲットデータベースに変換したスキーマを適用する場合、行った手動作業は上書きされます。

場合によっては、ターゲットデータベースにおいて同等のスキーマを作成できないことがあります。ターゲットデータベース向けにエンジンで利用できる機能を使用するために、アプリケーションとデータベースの一部を再設計することが必要な場合があります。その他の場合、自動的に変換できないスキーマは無視できます。

AWS SCT で変換されたスキーマの更新および再読み込み

AWS Schema Conversion Tool プロジェクトのソーススキーマおよびターゲットスキーマの両方を更新できます。

- [Source] (ソース) - ソースデータベースのスキーマを更新した場合、AWS SCT はプロジェクトのスキーマをソースデータベースの最新のスキーマと置き換えます。この機能を使用すると、ソースデータベースのスキーマが変更された場合にプロジェクトを更新できます。
- [Target] (ターゲット) - ターゲットデータベースのスキーマを更新した場合、AWS SCT はプロジェクトのスキーマをターゲットデータベースの最新のスキーマと置き換えます。ターゲットデータベースにスキーマを適用しなかった場合は、AWS SCT はプロジェクトから変換されたスキーマをクリアします。その後、クリーンターゲットデータベースのソースデータベースからスキーマを変換します。

AWS SCT プロジェクトでスキーマを更新するには、[Refresh from database] (データベースから更新) を選択します。

AWS SCT で変換されたスキーマの保存および適用

AWS Schema Conversion Tool が変換されたスキーマを生成する場合 (AWS SCT を使用したスキーマの変換 で示すように)、変換されたスキーマはターゲットデータベースにはすぐに適用されません。代わりに、変換されたスキーマは、ターゲットデータベースに適用する準備ができるまで、プロジェクトにローカルで保存されます。この機能を使用すると、ターゲットデータベースエンジンに自動的に変換できないスキーマ項目を使用できます。自動的に変換できない項目の詳細については、AWS SCT を使用した移行評価レポートの作成 を参照してください。

オプションとして、ターゲットデータベースにスキーマを適用する前に、変換されたスキーマをツールで SQL スクリプトとしてファイルに保存することができます。さらにツールで、変換したスキーマをターゲットデータベースに直接適用することもできます。

変換されたスキーマのファイルへの保存

変換されたスキーマを SQL スクリプトとしてテキストファイルに保存できます。これにより、ツールが自動的に変換できない項目に対応するために、AWS SCT から生成された SQL スクリプトを変更できます。ターゲットデータベースに変換されたスキーマを適用するために、ターゲット DB インスタンスで更新されたスクリプトを実行できます。

変換されたスキーマを SQL スクリプトとして保存するには

1. スキーマを選択して、コンテキスト (右クリック) メニューを開きます。
2. [Save as SQL] (SQL として保存) を選択します。
3. ファイルの名前を入力し、[Save] (保存) を選択します。
4. 変換されたスキーマは、次のいずれかのオプションを使用して保存できます。

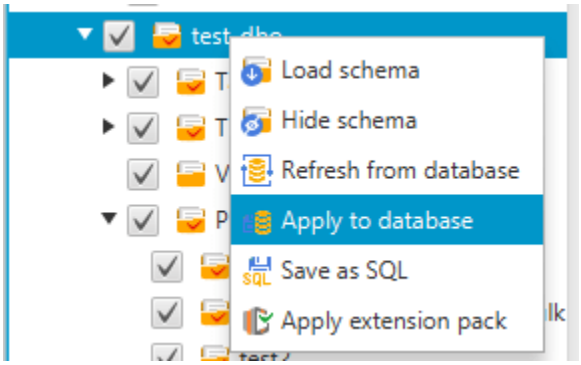
   - [単一ファイル]
   - [Single file per stage] (ステージごとに 1 つのファイル)
   - [Single file per statement] (ステートメントごとに 1 つのファイル)

SQL スクリプトの形式を選択するには

1. [Settings] (設定) メニューから [Project settings] (プロジェクト設定) を選択します。
2. [Save scripts] (スクリプトを保存) を選択します。
3. [Vendor] (ベンダー) で、データベースプラットフォームを選択します。
4. [Save SQL scripts to] (SQL スクリプトの保存先) で、データベーススキーマスクリプトの保存方法を選択します。
5. [OK] を選択して、設定を適用します。

変換されたスキーマの適用

変換されたスキーマをターゲットデータベースに適用する準備ができたら、プロジェクトの右側のパネルからスキーマ要素を選択します。スキーマ要素のコンテキスト (右クリック) メニューを開き、次に示すように、[Apply to database] (データベースに適用) を選択します。



拡張パックスキーマ

変換後のスキーマをターゲット DB インスタンスに適用すると、AWS SCT によって追加ワークスキーマがターゲット DB インスタンスに追加されます。このスキーマは、変換されたスキーマをターゲット DB インスタンスに書き込むときに必要なソースデータベースのシステム関数を実装します。追加されたスキーマは、拡張パックスキーマと呼ばれます。

この拡張パックスキーマは変更しないでください。変更すると、ターゲット DB インスタンスに書き込まれる変換されたスキーマに予期しない結果が発生する可能性があります。スキーマがターゲット DB インスタンスに完全に移行され、AWS SCT が必要なくなった場合は、拡張パックスキーマを削除できます。

拡張パックスキーマの名前は、ソースデータベースに従って次のように記述されます。

- Greenplum: `aws_greenplum_ext`
- Microsoft SQL Server: `aws_sqlserver_ext`
- Netezza: `aws_netezza_ext`
- Oracle: `aws_oracle_ext`
- Snowflake: `aws_snowflake_ext`
- Teradata: `aws_teradata_ext`
- Vertica: `aws_vertica_ext`

詳細については、「AWS SCT 拡張パックの使用」を参照してください。

Python ライブラリ

Amazon Redshift でカスタム関数を作成するには、Python 言語を使用します。AWS SCT 拡張子パックを使用して、python ライブラリを Amazon Redshift データベースにインストールします。詳細については、「AWS SCT 拡張パックの使用」を参照してください。

AWS SCT を使用した Amazon Redshift の最適化

Amazon Redshift データベースを最適化するには、AWS Schema Conversion Tool を使用することができます。Amazon Redshift データベースをソースとして、テスト用の Amazon Redshift データベースをターゲットとして使用する場合、AWS SCT では、データベースを最適化するための並べ替えキーおよびディストリビューションキーが推奨されます。

Amazon Redshift データベースの最適化

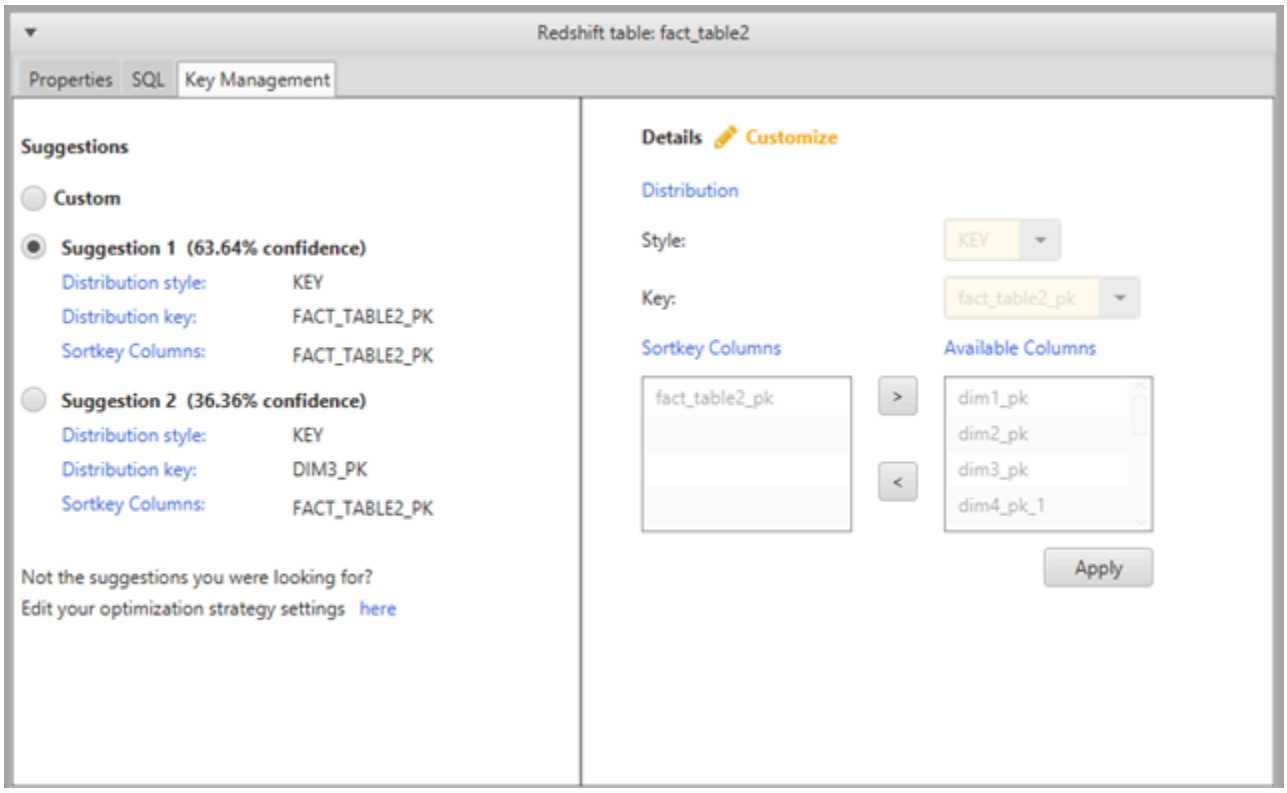
次の手順に従い、Amazon Redshift データベースを最適化します。

Amazon Redshift データベースを最適化するには

1. バックアップとして、マニュアルで Amazon Redshift クラスターのスナップショットを撮ります。Amazon Redshift クラスターを最適化し、変更のテストが完了したら、スナップショットを削除できます。詳細については、「Amazon Redshift スナップショット」を参照してください。
2. プロジェクトの左側のパネルから、変換するスキーマオブジェクトを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[Collect Statistics] (統計を収集) を選択します。

   AWS SCT は統計を使用して、並べ替えキーとディストリビューションキーを提案します。
3. プロジェクトの左側のパネルから、最適化するスキーマオブジェクトを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[Run Optimization] (最適化を実行) を選択します。

   AWS SCT は並べ替えキーとディストリビューションキーを提案します。
4. 提案を確認するには、プロジェクトの左側のパネルのスキーマにあるテーブルのノードを展開し、テーブルを選択します。次に示されているように、[Key Management] (キーの管理) タブを選択します。



左側のペインにはキーの提案が含まれ、各提案の信頼性に関する評価が含まれています。いずれかの提案を選択するか、右側のペインで編集してキーをカスタマイズできます。

5. 最適化の提案が記載されているレポートを作成できます。レポートを作成するには、次の手順に従います。

   a. プロジェクトの左側のパネルから、最適化したスキーマオブジェクトを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[Create Report] (レポートを作成) を選択します。

      メインウィンドウでレポートが開き、[Summary] (概要) タブが表示されます。最適化の提案があるオブジェクトの数がレポートに記載されます。

   b. [Action Items] (アクション項目) タブを選択して、レポートフォーマットにある主要な提案を確認します。

   c. 最適化レポートのローカルコピーは PDF ファイルまたはカンマ区切り値 (CSV) ファイルとして保存できます。CSV ファイルには、アクション項目の情報のみが含まれます。PDF ファイルには、要約とアクション項目の情報の両方が含まれます。

6. 提案された最適化をデータベースに適用するには、プロジェクトの右側パネルでオブジェクトを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[Apply to database] (データベースに適用) を選択します。

AWS Schema Conversion Tool を使用した抽出、変換、ロード (ETL) プロセスの変換

AWS Schema Conversion Tool (AWS SCT) を使用して、抽出、変換、ロード (ETL) プロセスを移行できます。この移行タイプには、ETL 関連のビジネスロジックの変換も含まれます。このロジックは、ソースデータウェアハウス内や、個別に実行された、外部スクリプト内に配置できます。

現在、AWS SCT は、次の表に示すように、ETL スクリプトから AWS Glue と Amazon Redshift RSQL へのオブジェクトへの変換をサポートしています。

ソース	ターゲット
Informatica ETL スクリプト	Informatica
Microsoft SQL Server Integration Services (SSIS) ETL パッケージ	AWS Glue、または AWS Glue Studio
Teradata Basic Teradata Query (BTEQ) からのコマンドが埋め込まれた、シェルスクリプト	Amazon Redshift RSQL
テラデータ BTEQ ETL スクリプト	AWS Glue またはAmazon Redshift RSQL
Teradata FastExport ジョブスクリプト	Amazon Redshift RSQL
Teradata FastLoad ジョブスクリプト	Amazon Redshift RSQL
Teradata MultiLoad ジョブスクリプト	Amazon Redshift RSQL

トピック

- AWS SCT を使用して、Teradata BTEQ コマンドが埋め込まれているシェルスクリプトを Amazon Redshift RSQL に変換する
- AWS SCT を使用してた Teradata FastExport ジョブスクリプトの Amazon Redshift RSQL への変換
- AWS SCT を使用した Teradata FastLoad ジョブスクリプトの Amazon Redshift RSQL への変換
- AWS SCT でTeradata MultiLoad ジョブスクリプトから Amazon Redshift RSQL への変換を行う

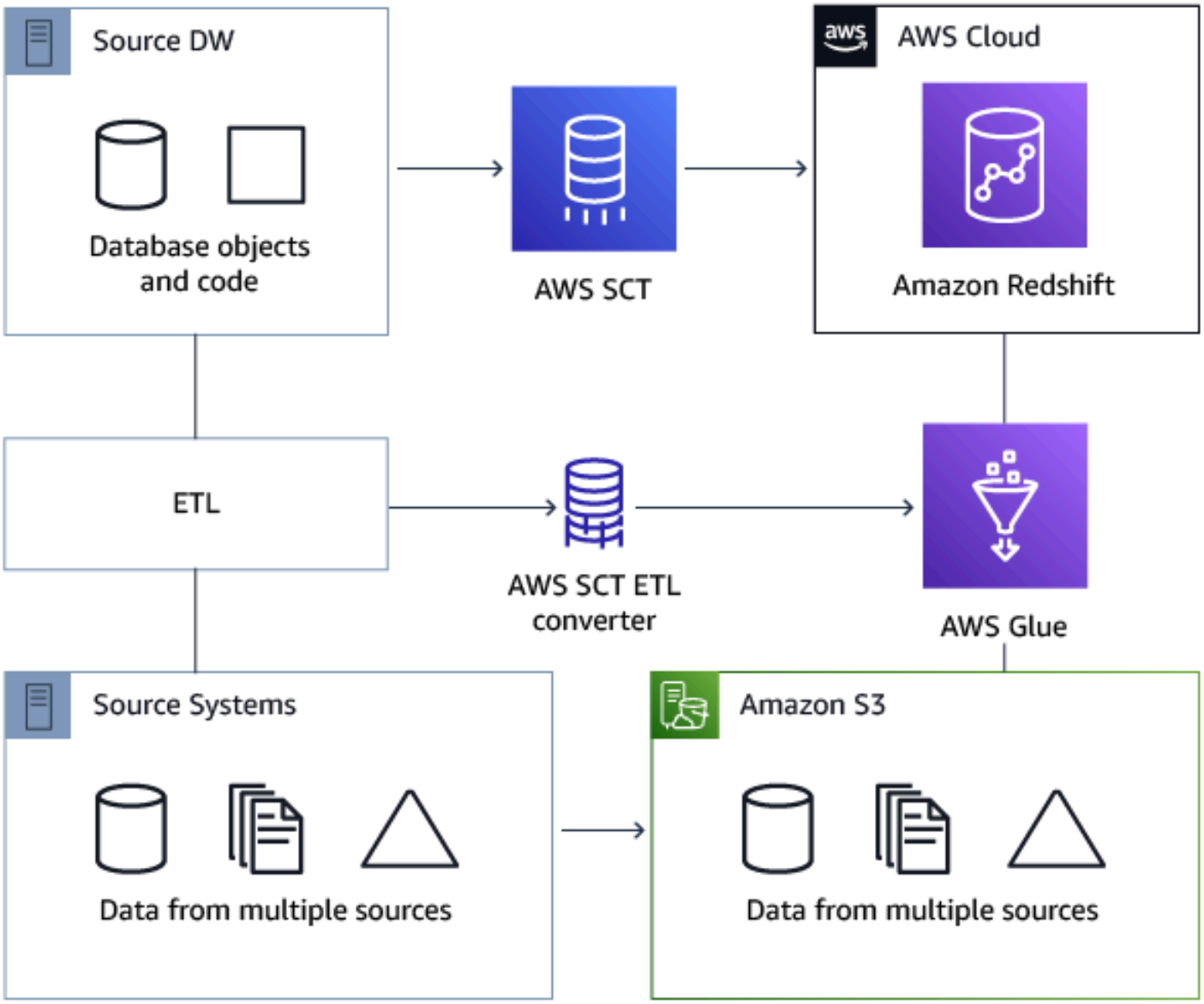
AWS SCT を使用して ETL プロセスを AWS Glue に変換する

以下のセクションでは、AWS SCT を使用して ETL スクリプトを AWS Glue に変換するプロセスの概要を示しています。この例では、ソースデータベースおよびデータウェアハウスに使用される ETL プロセスとともに、Oracle データベースから Amazon Redshift への変換を行います。

トピック

- 前提条件
- AWS Glue データカタログについて
- AWS Glue で AWS SCT を使用した変換の制限
- ステップ 1: 新しい プロジェクトを作成する
- ステップ 2: AWS Glue ジョブを作成する

次のアーキテクチャ図は、ETL スクリプトを AWS Glue に変換するデータベース移行プロジェクトの例を示しています。



前提条件

開始する前に、以下を実行します。

- AWS に移行する予定のソースデータベースをすべて移行します。
- ターゲットデータウェアハウスを AWS に移行します。
- ETL プロセスに関わるすべてのコードのリストを収集します。
- 各データベースに必要なすべての接続情報のリストを収集します。

ユーザーの代わりに AWS Glue は他の AWS リソースにアクセスするためのアクセス許可が必要です。AWS Identity and Access Management (IAM) を使用してアクセス権限を提供できます。必ず、AWS Glue の IAM ポリシーを作成してください。詳細については、『AWS Glue 開発者ガイド』の「AWS Glueservice 用の「IAM ポリシーの作成」を参照してください。

AWS Glue データカタログについて

変換プロセスの一環として、AWS Glue にはソースデータベースとターゲットデータベースに関する情報が読み込まれます。この情報は、ツリーと呼ばれる構造で各種カテゴリに分類されます。この構造には、以下のものが含まれています。

- 接続 - 接続パラメータ
- クローラー - クローラーのリスト。スキーマごとに 1 つのクローラーが割り当てられます。
- データベース - テーブルを保持するコンテナ
- テーブル - テーブル内のデータを表すメタデータ定義
- ETL ジョブ - ETL 操作を実行するビジネスロジック
- トリガー - AWS Glue で ETL ジョブを実行するタイミング (オンデマンド、スケジュール、ジョブイベントによるトリガーなど) を制御するロジック

AWS Glue データカタログは、データの場所、スキーマ、およびランタイムメトリクスへのインデックスです。AWS Glue および AWS SCT を使用する場合、 AWS Glue データカタログには AWS Glue の ETL ジョブのソースおよびターゲットとして使用されるデータへのリファレンスが含まれます。データウェアハウスを作成するには、このデータを分類します。

データカタログ内の情報は、ETL ジョブの作成と監視に使用します。一般的には、クローラーを実行してデータストア内のデータのインベントリを行いますが、データカタログにメタデータテーブルを追加する別の方法もあります。

データカタログでテーブルを定義したら、データベースに追加します。データベースは AWS Glue でテーブルを整理するために使用されます。

AWS Glue で AWS SCT を使用した変換の制限

AWS Glue では、AWS SCT を使用して変換する場合に以下の制限が適用されます。

リソース	デフォルトの制限

アカウントあたりのデータベース数	10,000
データベースあたりのテーブル数	100,000
テーブルあたりのパーティションの数	1,000,000
テーブルあたりのテーブルバージョンの数	100,000
アカウントあたりのデータベース数	1,000,000
アカウントあたりのパーティションの数	10,000,000
アカウントあたりのテーブルバージョンの数	1,000,000
アカウントあたりの接続数	1,000
アカウントあたりのクローラー数	25
アカウントあたりのジョブの数	25
アカウントあたりのトリガー数	25
アカウントあたりの同時ジョブの実行数	30
ジョブあたりの同時ジョブの実行数	3
トリガーごとのジョブ数	10
アカウントごとの開発エンドポイントの数	5
開発エンドポイントによって一度に使用される最大データ処理単位 (DPU) 数	5
一度にロールによって使用される最大 DPU 数	100

データベース名の長さ	無制限 Apache Hive など、その他のメタデータストアとの互換性を考慮して、名前は小文字に変換されます。 Amazon Athena からデータベースにアクセスする場合は、英数字とアンダースコア文字のみを使用して名前を指定してください。
接続名の長さ	無制限
クローラー名の長さ	無制限

ステップ 1: 新しい プロジェクトを作成する

新しいプロジェクトを作成するには、以下の大まかな手順を実行します。

1. AWS SCT で新しいプロジェクトを作成します。詳細については、「AWS SCT プロジェクトの作成」を参照してください。
2. ソースデータベースとターゲットデータベースをプロジェクトに追加します。詳細については、「AWS SCT プロジェクトへのデータベースサーバーの追加」を参照してください。

   ターゲットデータベースの接続設定で [AWS Glue の使用] を選択していることを確認してください。それには、[AWS Glue] タブを選択します。[AWS プロファイルからコピー] から、使用するプロファイルを選択します。プロファイルの内容が、AWS アクセスキー、シークレットキー、および Amazon S3 バケットフォルダーのフィールドに自動的に表示されます。表示されない場合は、この情報を手動で入力します。[OK] を選択すると、AWS Glue によってオブジェクトの分析が行われ、メタデータは AWS Glue データカタログにロードされます。

   セキュリティ設定によっては、サーバー上の一部のスキーマに対する十分な権限がアカウントにないことを示す警告メッセージが表示される場合があります。使用するスキーマへのアクセス権がある場合は、このメッセージを無視しても問題ありません。
3. ETL をインポートする準備を完了するには、ソース データベースとターゲット データベースに接続します。そのためには、ソースまたはターゲットのメタデータツリーでデータベースを選択し、[サーバーに接続] を選択します。

AWS Glue により、ETL 変換用のデータベースがソースデータベースサーバーとターゲットデータベースサーバーに作成されます。ターゲットサーバーのデータベースには AWS Glue データカタログが格納されます。特定のオブジェクトを見つけるには、ソースまたはターゲットパネルの検索を使用します。

特定のオブジェクトの変換方法を確認するには、変換する項目を指定し、コンテキスト (右クリック) メニューから [スキーマの変換] を選択します。AWS SCT によって、スクリプトに変換されます。

変換されたスクリプトは、右側のパネルの [Scripts] (スクリプト) フォルダを選択します。現在、スクリプトは仮想オブジェクトであり、このスクリプトは、AWS SCT プロジェクトの一部としてのみ利用できます。

変換したスクリプトで AWS Glue ジョブを作成するには、スクリプトを Amazon S3 にアップロードします。スクリプトを Amazon S3 にアップロードするには、スクリプトを選択し、コンテキスト (右クリック) メニューから [S3 に保存] を選択します。

ステップ 2: AWS Glue ジョブを作成する

スクリプトを Amazon S3 に保存したら、そのスクリプトを選択して [AWS Glue ジョブの設定] を選択すると、AWS Glue ジョブを設定するためのウィザードが開きます。ウィザードを使用すると、設定を簡単に行うことができます。

1. ウィザードの最初のタブの [設計データフロー] で、実行戦略と、このジョブに含めるスクリプトのリストを選択します。各スクリプトのパラメータも選択できます。また、正しい順序で実行されるようにスクリプトを並べ替えることもできます。
2. 2 番目のタブでは、ジョブに名前を付け、AWS Glue の設定を直接設定できます。この画面では、次の設定を指定できます。
   - AWS Identity and Access Management (IAM) ロール
   - スクリプトファイルの名前とファイルパス
   - Amazon S3 が管理するキーによるサーバー側の暗号化 (SSE-S3) を使用したスクリプトの暗号化
   - 一時ディレクトリ
   - 生成された Python ライブラリパス
   - ユーザーの Python ライブラリパス
   - 依存する.jar ファイルのパス
   - 参照されるファイルパス
   - 実行されるジョブごとの同時 DPU

- 最大同時実行数
- ジョブのタイムアウト (分)
- 遅延通知のしきい値 (分)
- 再試行回数
- セキュリティ設定
- サーバー側の暗号化

3. 3 番目のステップ (タブ) では、ターゲットエンドポイントへの設定済みの接続を選択します。

設定が完了したジョブは、AWS Glue データカタログ内の ETL ジョブの下に表示されます。ジョブを選択すると設定が表示され、確認または編集を行うことができます。AWS Glue で新しいジョブを作成するには、ジョブのコンテキスト (右クリック) メニューから [AWS Glue ジョブの作成] を選択します。これにより、スキーマ定義が適用されます。表示を更新するには、コンテキスト (右クリック) メニューから [Refresh from database] (データベースから更新) を選択します。

この時点で、AWS Glue コンソールにジョブを表示できます。これを行うには、AWS Management Console にサインインし、AWS Glue コンソール (https://console.aws.amazon.com/glue/) を開きます。

新しいジョブをテストして、正常に動作していることを確認します。そのためには、まずソーステーブルのデータを確認し、ターゲットテーブルが空であることを確認します。ジョブを実行してから、もう一度確認します。エラーログは AWS Glue コンソールから表示できます。

AWS SCT で AWS Glue の Python API を使用して ETL プロセスを変換する

以下のセクションでは、Python で AWS Glue API オペレーションを呼び出す変換について説明します。詳細については、AWS Glue 開発者ガイド の「Python での AWS Glue ETL スクリプト」を参照してください。

トピック

- ステップ 1: データベースを作成する
- ステップ 2: 接続を作成する
- ステップ 3: AWS Glue クローラーの作成

ステップ 1: データベースを作成する

最初のステップでは、AWS SDK API を使用して、AWS Glue データカタログに新しいデータベースを作成します。データカタログに定義したテーブルは、データベースに追加されます。データベースは、AWS Glue でテーブルを整理するために使用されます。

次の例は、AWS Glue の Python API の `create_database` メソッドを示しています。

```
response = client.create_database(
    DatabaseInput={
        'Name': 'database_name',
        'Description': 'description',
        'LocationUri': 'string',
        'Parameters': {
            'parameter-name': 'parameter value'
        }
    }
)
```

Amazon Redshift を使用している場合のデータベース名は次のとおりです。

```
{redshift_cluster_name}_{redshift_database_name}_{redshift_schema_name}
```

この例では、Amazon Redshift クラスターのフルネームは次のようになります。

```
rsdbb03.apq1mpqso.us-west-2.redshift.amazonaws.com
```

正しい形式のデータベース名の例を以下に示します。この場合は、`rsdbb03` のようになります。これは、クラスターエンドポイントのフルネームの最初の部分です。データベースの名前は `dev` で、スキーマは `ora_glue`です。

```
rsdbb03_dev_ora_glue
```

ステップ 2: 接続を作成する

AWS SDK API を使用して、データカタログに新しい接続を作成します。

次の例は、AWS Glue の Python API の `create_connection` メソッドを使用した例を示しています。

```
response = client.create_connection(
    ConnectionInput={
        'Name': 'Redshift_abcde03.aabbcc112233.us-west-2.redshift.amazonaws.com_dev',
        'Description': 'Created from SCT',
        'ConnectionType': 'JDBC',
        'ConnectionProperties': {
            'JDBC_CONNECTION_URL': 'jdbc:redshift://aabbcc03.aabbcc112233.us-
west-2.redshift.amazonaws.com:5439/dev',
            'USERNAME': 'user_name',
            'PASSWORD': 'password'
        },
        'PhysicalConnectionRequirements': {
            'AvailabilityZone': 'us-west-2c',
            'SubnetId': 'subnet-a1b23c45',
            'SecurityGroupIdList': [
                'sg-000a2b3c', 'sg-1a230b4c', 'sg-aba12c3d', 'sg-1abb2345'
            ]
        }
    }
)
```

`create_connection` で使用されているパラメータは次のとおりです。

- Name (UTF-8 文字列) - 必須。Amazon Redshift では、接続名は次のような形式になります: `Redshift_<Endpoint-name>_<redshift-database-name>` (例: `Redshift_abcde03_dev`)
- `Description` (UTF-8 文字列) - 接続の説明。
- `ConnectionType` (UTF-8 文字列) - 必須。接続のタイプ。現時点では JDBC のみがサポートされており、SFTP はサポート外です。
- `ConnectionProperties` (dict) – 必須。この接続のパラメータとして使用されるキーと値のペアのリスト (例: JDBC 接続 URL、ユーザー名、およびパスワード)。
- `PhysicalConnectionRequirements` (dict) - 物理接続の要件。以下の内容が含まれます。
  - `SubnetId` (UTF-8 文字列) - 接続で使用されるサブネットの ID。
  - `SecurityGroupIdList` (list) - 接続で使用されるセキュリティグループ ID リスト。
  - `AvailabilityZone` (UTF-8 文字列) - 必須。そのエンドポイントを含むアベイラビリティーゾーン。このパラメータは廃止されました。

ステップ 3: AWS Glue クローラーの作成

次に、AWS Glue クローラーを作成して、AWS Glue カタログを追加します。詳細については、『AWS Glue デベロッパーガイド』の「クローラーを使用したデータのカタログ化」を参照してください。

クローラーの作成における最初のステップでは、AWS SDK API を使用して、データカタログに新しいデータベースを作成します。開始する前にまず、delete_crawler オペレーションを使用して、以前のバージョンをすべて削除する必要があります。

クローラーを作成するときは、いくつかの考慮事項が適用されます。

- クローラー名には、*<redshift_node_name>_<redshift_database_name>_<redshift_shema_name>* 形式を使用します (例: abcde03_dev_ora_glue)。
- 既存の IAM ロールを使用します。IAM ロールの作成の詳細については、『IAM ユーザーガイド』の「IAM ロールの作成」を参照してください。
- 前のステップで作成したデータベースの名前を使用します。
- ConnectionName パラメータ (必須) を使用します。
- path パラメータでは、JDBC ターゲットへのパス (例: dev/ora_glue/%) を使用します。

次の例では、既存のクローラーを削除し、AWS Glue の Python API を使用して新しいクローラーを作成します。

```
response = client.delete_crawler(
    Name='crawler_name'
)

response = client.create_crawler(
    Name='crawler_name',
    Role= 'IAM_role',
    DatabaseName='database_name',
    Description='string',
    Targets={
        'S3Targets': [
            {
                'Path': 'string',
                'Exclusions': [
                    'string',
```

```
                ]
            },
        ],
        'JdbcTargets': [
            {
                'ConnectionName': 'ConnectionName',
                'Path': 'Include_path',
                'Exclusions': [
                    'string',
                ]
            },
        ]
    },
    Schedule='string',
    Classifiers=[
        'string',
    ],
    TablePrefix='string',
    SchemaChangePolicy={
        'UpdateBehavior': 'LOG'|'UPDATE_IN_DATABASE',
        'DeleteBehavior': 'LOG'|'DELETE_FROM_DATABASE'|'DEPRECATE_IN_DATABASE'
    },
    Configuration='string'
)
```

1 つ以上のデータストアに接続してデータ構造を決定し、データカタログにテーブルを書き込むクローラーを作成して、実行します。スケジュールに基づいてクローラーを実行することができます (以下参照)。

```
response = client.start_crawler(
    Name='string'
)
```

この例では、Amazon Redshift をターゲットとして使用しています。Amazon Redshift AWS Glue データ型は、クローラーの実行後に次の方法でデータ型にマッピングされます。

Amazon Redshift のデータ型	AWS Glue データ型
smallint	smallint
integer	整数

bigint	bigint
decimal	decimal(18,0)
decimal(p,s)	decimal(p,s)
real	double
double precision	double
boolean	boolean
char	文字列
varchar	文字列
varchar(n)	文字列
date	date
timestamp	timestamp
timestamptz	timestamp

AWS SCT による Informatica ETL スクリプトの変換

AWS SCT コマンドラインインターフェイス (CLI) を使用して Informatica ETL スクリプトを変換し、そのスクリプトを新しいターゲットデータベースで使用できるようにすることができます。この変換には 3 つの重要なステップが含まれます。まず、AWS SCT は Informatica オブジェクトに埋め込まれている SQL コードを変換します。次に、AWS SCT はプロジェクトで指定した移行ルールに従ってデータベースオブジェクトの名前を変更します。最後に、AWS SCT は Informatica ETL スクリプトの接続を新しいターゲットデータベースにリダイレクトします。

Informatica ETL スクリプトは、AWS SCT データベース変換プロジェクトの一部として変換できます。Informatica ETL スクリプトを変換するときは、必ずソースデータベースとターゲットデータベースをプロジェクトに追加してください。

インフォマティカ ETL スクリプトを変換するには、必ず AWS SCT バージョン 1.0.667 以降を使用してください。また、AWS SCT のコマンドラインインターフェイスにも慣れておいてください。詳細については、「AWS SCT CLI リファレンス」を参照してください。

AWS SCT を使用して Informatica ETL スクリプトを変換するには

1. 新しい AWS SCT CLI スクリプトを作成するか、既存のシナリオテンプレートを編集します。例えば、InformaticConversionTemplate.scts テンプレートをダウンロードして編集できます。詳細については、「CLI シナリオの取得」を参照してください。
2. ソースデータベースとターゲットデータベースに必要な JDBC ドライバーをダウンロードします。コマンドを使用して、これらのドライバーの場所を指定します。SetGlobalSettingsまた、AWS SCT がログファイルを保存できるフォルダも指定します。

   次のコード例では、Oracle ドライバーと PostgreSQL ドライバーのパスを AWS SCT 設定に追加する方法を示します。このコード例を実行すると、AWS SCT はログファイルをフォルダに保存します。C:\sct_logまた、AWS SCT は コンソールログファイルも C:\Temp\oracle_postgresql フォルダに保存します。

```
SetGlobalSettings
 -save: 'true'
 -settings: '{"oracle_driver_file": "C:\\drivers\\ojdbc8.jar",
 "postgresql_driver_file": "C:\\drivers\\postgresql-42.2.19.jar" }'
/

SetGlobalSettings
 -save: 'false'
 -settings: '{
"log_folder": "C:\\sct_log",
"console_log_folder": "C:\\Temp\\oracle_postgresql"}'
/
```

3. 新しい AWS SCT プロジェクトを作成します。プロジェクトの名前と場所を入力します。

   次のコード例では、C:\Temp フォルダに oracle_postgresql プロジェクトを作成します。

```
CreateProject
 -name: 'oracle_postgresql'
 -directory: 'C:\Temp'
/
```

4. ソースデータベースとターゲットデータベースに関する接続情報を追加します。

   次のコード例では、Oracle データベースと PostgreSQL データベースを AWS SCT プロジェクトのソースおよびターゲットとして追加します。

```
AddSource
 -password: 'source_password'
 -port: '1521'
 -vendor: 'ORACLE'
 -name: 'ORACLE'
 -host: 'source_address'
 -database: 'ORCL'
 -user: 'source_user'
/
AddTarget
 -database: 'postgresql'
 -password: 'target_password'
 -port: '5432'
 -vendor: 'POSTGRESQL'
 -name: 'POSTGRESQL'
 -host: 'target_address'
 -user: 'target_user'
/
```

前の例では、*source_user* と *target_user* をデータベースユーザーの名前に置き換えてください。次に、*source_password* と *target_password* を自分のパスワードに置き換えます。*source_address* と *target_address* には、ソースデータベースサーバーとターゲットデータベースサーバーの IP アドレスを入力します。

バージョン 19 以降の Oracle データベースに接続するには、AddSource コマンドに Oracle サービス名を使用します。そのためには、-connectionType パラメータを追加し、値を 'basic_service_name' に設定します。次に、-servicename パラメータを追加し、その値を Oracle サービス名に設定します。AddSource コマンドの使用の詳細については、『AWS Schema Conversion Tool CLI リファレンス』を参照してください。

5. 各ソースデータベーススキーマのターゲットデータベースエンジンを定義する新しい AWS SCT マッピングルールを作成します。詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。

   次のコード例では、すべてのソース Oracle データベーススキーマを含むマッピングルールを作成し、PostgreSQL を移行ターゲットとして定義します。

```
AddServerMapping
 -sourceTreePath: 'Servers.ORACLE'
 -targetTreePath: 'Servers.POSTGRESQL'
```

```
/
```

6. Informatica のソースとターゲット XML ファイルに関する接続情報を追加します。

   次のコード例では、C:\Informatica_source および C:\Informatica_target フォルダから Informatica XML ファイルを追加します。

```
AddSource
 -name: 'INFA_SOURCE'
 -vendor: 'INFORMATICA'
 -mappingsFolder: 'C:\Informatica_source'
/
AddTarget
 -name: 'INFA_TARGET'
 -vendor: 'INFORMATICA'
 -mappingsFolder: 'C:\Informatica_target'
/
```

7. 別のマッピングルールを作成して、ソースの Informatica XML ファイル用のターゲット Informatica XML ファイルを定義します。

   次のコード例では、前の例で使用したソースとターゲットの Informatica XML ファイルを含むマッピングルールを作成します。

```
AddServerMapping
-sourceTreePath: 'ETL.INFA_SOURCE'
-targetTreePath: 'ETL.INFA_TARGET'
/
```

8. Informatica 接続名参照に対応するデータベースサーバー接続を指定します。

   次のコード例では、Informatica ETL スクリプトをソースから新しいターゲットデータベースにリダイレクトするように設定しています。この例では接続変数も設定しています。

```
ConfigureInformaticaConnectionsRedirect
 -treePath: 'ETL.INFA_SOURCE.Files'
 -connections: '{
 "ConnectionNames": [
 {
  "name": "Oracle_src",
  "newName": "postgres",
  "treePath": "Servers.ORACLE"
```

```
 }
 ]
 "ConnectionVariables": [
 {
         "name": "$Source",
         "treePath": "Servers.ORACLE"
    }
    ]
 }'
/
```

9. ソースデータベーススキーマと Informatica ETL スクリプトを変換します。

   次のコード例は、すべてのソース Oracle データベーススキーマと Informatica XML ファイルを変換します。

```
Convert
 -treePath: 'Servers.ORACLE.Schemas.%'
/
Convert
 -treePath: 'ETL.INFA_SOURCE.Files'
/
```

10. (オプション) 変換プロジェクトと評価レポートを保存します。このレポートには、変換アクション項目と、それぞれに対処する方法に関する推奨事項が含まれます。

    次のコード例では、プロジェクトを保存し、評価レポートのコピーを PDF ファイルとして C:\Temp フォルダに保存します。

```
SaveProject
/
SaveReportPDF
 -treePath: 'ETL.INFA_SOURCE.Files'
 -file:'C:\Temp\Informatica.pdf'
/
```

11. 変換した Informatica XML ファイルを保存します。

    次のコード例では、変換された XML C:\Temp ファイルをフォルダに保存します。このフォルダは、AddTarget前のステップでコマンドを使用して指定しました。

```
SaveTargetInformaticaXML
```

```
-treePath: 'ETL.INFA_TARGET.Files'
/
```

12. .scts スクリプトをファイルとして保存し、AWS SCT CLI RunSCTBatch のコマンドを使用して実行します。詳細については、「AWS SCT CLI スクリプトモード」を参照してください。

    次の例では、C:\Temp フォルダ で Informatica.scts スクリプトを実行します。この例は Windows でも使用できます。

    ```
    RunSCTBatch.cmd --pathtoscts "C:\Temp\Informatica.scts"
    ```

    ソースのInformatica ETLスクリプトを編集した場合は、AWS SCT CLI スクリプトを再度実行してください。

AWS SCT を使用して SSIS をAWS Glue に変換する

次で、AWS SCT を使用して Microsoft SQL Server Integration Services (SSIS) AWS Glue パッケージを変換する方法を確認することができます。

Microsoft SSIS パッケージを AWS Glue に変換するには、AWS SCT バージョン 1.0.642 以降を使用していることを確認してください。ETL パッケージを使用したお客様の SSIS プロジェクト — ローカルフォルダ内の .dtsx、.conmgr、および .params ファイル。

SSIS サーバーをインストールする必要はありません。変換プロセスは、ローカル SSIS ファイルを介して行われます。

AWS SCT を使用して SSIS パッケージを AWS Glue に変換するには

1. AWS SCT で新しいプロジェクトを作成するか、既存のプロジェクトを開きます。詳細については、「the section called “プロジェクトの作成”」を参照してください。
2. メニューから [ソースを追加] を選択し、新しいソース SSIS パッケージをプロジェクトに追加します。
3. [SQL Server 統合サービス] を選択し、以下を完了します。

   - [接続名] — 接続の名前を入力します。AWS SCT はこの名前をメタデータツリーに表示します。
   - [SSIS パッケージフォルダ] — パッケージを含む SSIS プロジェクトフォルダーのパスを選択します。

AWS SCT はローカルフォルダからプロジェクトファイル (拡張子の付いたファイル `.dtsx`、`.conmgr` または `.params`) をロード、解析し、 のカテゴリツリーに整理します。次に、それらを AWS SCT カテゴリツリーに整理します。

4. メニューから [ターゲットを追加] を選択し、ソース SSIS パッケージを変換する新しいターゲットプラットフォームを追加します。
5. [AWS Glue] を選択して、以下の手順を完了します。

   - [接続名] — 接続の名前を入力します。AWS SCT は、この名前をメタデータツリーに表示します。
   - [AWS プロファイルからコピー] — 使用するプロファイルを選択します。
   - [AWS アクセスキー] - AWS のアクセスキーを入力します。
   - [AWS シークレットキー] - AWS のシークレットキーを入力します。
   - [リージョン] — リストから AWS リージョン リージョンを選択します。
   - [Amazon S3 バケットフォルダ] — 使用する予定の Amazon S3 バケットのフォルダパスを入力します。

   仮想 AWS Glue ターゲットを使用できます。この場合、接続認証情報を指定する必要はありません。詳細については、「the section called “仮想ターゲット”」を参照してください。
6. ソース SSIS パッケージと AWS Glue ターゲットを含む新しいマッピングルールを作成します。詳細については、「the section called “新しいルール”」を参照してください。
7. [ビュー] メニューで、[メインビュー] を選択します。
8. SSIS ツリービューで、[Connection managers] (接続マネージャ) コンテキスト (右クリック) メニューを開き、[Configure connections] (接続の設定) を選択します。
9. プロジェクト接続マネージャを設定します。

   SSIS 接続マネージャの接続マッピングを設定するには、対応する SSIS 接続マネージャの AWS Glue 接続を指定します。AWS Glue 接続がすでに作成されていることを確認します。

   a. [Connections] (接続) で、[Project connection] (プロジェクト接続) を選択します。
   b. [Glue カタログ接続] で、適切な AWS Glue 接続を選択します。
10. パッケージ接続マネージャを設定します。

    a. [接続] で、パッケージを選択します。

b. [Glue カタログ接続] で、適切な AWS Glue 接続を選択します。

c. パッケージで使用可能なすべての接続について、これらのアクションを繰り返します。

11. [Apply (適用)] を選択します。
12. パッケージを変換します。ソースツリービューで、[パッケージ] を探します。パッケージのコンテキスト (右クリック) メニューを開き、[パッケージの変換] を選択します。
13. 変換されたスクリプトを Amazon S3 に保存します。ターゲットのツリービューで、[パッケージスクリプト] を見つけます。変換されたスクリプトのコンテキスト (右クリック) メニューを開き、[S3 に保存] を選択します。
14. AWS Glue ジョブを設定します。ターゲットのツリービューで、[Package スクリプト] を見つけます。変換したスクリプトのコンテキスト (右クリック) メニューを開き、[AWS Glue ジョブの設定] を選択します。
15. 次に示す、3 つの設定セクションを完了します。

    a. [デザインデータフロー] セクションに入力します。

    - [実行戦略] — ジョブで ETL スクリプトを実行する方法を選択します。[シーケンシャル] を選択すると、ウィザードで指定した順序でスクリプトが実行されます。[並行] を選択すると、ウィザードで指定された順序を無視してスクリプトが並行して実行されます。
    - [スクリプト] — 変換したスクリプトの名前を選択します。
    - [次へ] をクリックします。

    b. [Job プロパティ]セクションに入力します。

    - [名前] – AWS Glue ジョブの名前を入力します。
    - [IAM ロール] – ジョブの実行とデータストアへのアクセスに使用されるリソースの認証に使用する IAM ロールを選択します。
    - [スクリプトファイル名] — 変換したスクリプトの名前を入力します。
    - [スクリプトファイル S3 パス] — 変換されたスクリプトへの Amazon S3 パスを入力します。
    - [SSE-S3 を使用してスクリプトを暗号化] — Amazon S3 が管理する暗号化キー (SSE-S3) によるサーバー側の暗号化を使用してデータを保護するには、このオプションを選択します。
    - [一時ディレクトリ] — 中間結果用の一時ディレクトリへの Amazon S3 パスを入力します。AWS Glue および AWS Glue 組み込みトランスフォームは、このディレクトリを使用して Amazon Redshift への読み取りまたは書き込みを行います。

- AWS SCT は、Python ライブラリのパスを自動的に生成します。このパスは [生成された Python ライブラリパス] で確認できます。この自動生成されたパスは編集できません。追加の Python ライブラリを使用するには、[ユーザ Python ライブラリパス] にパスを入力します。
- [ユーザ Python ライブラリパス] — 追加のユーザ Python ライブラリのパスを入力します。Amazon S3 のパスはカンマで区切ります。
- [依存 jars ファイルのパス] – 依存 .jars ファイルのパスを入力します。Amazon S3 のパスはカンマで区切ります。
- [参照ファイルパス] — 設定ファイルなど、スクリプトに必要な追加ファイルのパスを入力します。Amazon S3 のパスはカンマで区切ります。
- [最大キャパシティー] – 最大キャパシティーは、このジョブの実行時に割り当てることができる AWS Glue データ処理ユニット (DPU) の数です。2 から 100 の任意の整数を入力できます。デフォルトは 2 です。
- [同時実行の最大数] – このジョブで許可される同時実行の最大数を設定します。デフォルトは 1 です。このしきい値に達すると、AWS Glue はエラーを返します。
- [Job タイムアウト (分)] — ジョブの暴走を防ぐため、ETL ジョブのタイムアウト値を入力します。バッチジョブのデフォルト値は 2880 分 (48 時間) です。ジョブがこの制限を超えると、ジョブの実行状態は `TIMEOUT` に変わります。
- [遅延通知のしきい値 (分)] — AWS SCT が遅延通知を送信するまでのしきい値を分単位で入力します。
- [リトライ回数] — AWS Glue が失敗した場合にジョブを自動的に再開する回数 (0 ~ 10 回) を入力します。タイムアウト制限に達したジョブは再起動されません。デフォルトは 0 です。
- [次へ] をクリックします。

c. 必要な接続を設定します。

   i. [すべての接続] AWS Glue から必要な接続を選択し、[選択した接続」のリストに追加します。

   ii. [終了] を選択します。

16. 設定済みの AWS Glue ジョブを作成します。ターゲットのツリービューで、[ETL ジョブ] を見つけて展開します。設定した ETL ジョブのコンテキスト (右クリック) メニューを開き、[AWS Glue ジョブの作成] を選択します。
17. AWS Glue ジョブを実行します。

a. https://console.aws.amazon.com/glue/ で AWS Glue コンソール を開きます。
b. ナビゲーションペインで [Jobs] (ジョブ) を選択します。
c. [ジョブの追加] タブで、実行するジョブを選択します。
d. [Action] (アクション) タブで、[Run job] (ジョブの実行) を選択します。

AWS SCT が AWS Glue に変換できる SSIS コンポーネント

AWS SCT を使用して、データフローコンポーネントと制御フローコンポーネントのほか、コンテナ、パラメータ、変数を変換できます。

サポートされるデータフローコンポーネントは次のとおりです。

- ADO NET 送信先
- ADO NET 送信元
- 集計
- キャッシュ変換
- キャラクタマップ変換
- 条件付き分割変換
- コピー列変換
- データ変換
- 派生列変換
- Excel 送信先
- Excel 送信元
- エクスポート列変換
- フラットファイル送信先
- フラットファイルソース
- ファジー検索変換
- インポート列変換
- 参照変換
- マージ結合変換
- マージ変換

- マルチキャスト変換
- ODBC 送信先
- ODBC 送信元
- OLE DB コマンド変換
- OLE DB 送信先
- OLE DB 送信元
- パーセンテージサンプリング変換
- ピボット変換
- raw ファイル送信先
- raw ファイル送信元
- RecordSet 送信先
- 行数変換
- 行サンプリング変換
- ソート変換
- SQL Server 送信先
- Union All 変換
- ピボット解除変換
- XML ソース

サポートされている制御フローコンポーネントは以下のとおりです。

- 一括挿入タスク
- パッケージ実行タスク
- SQL 実行タスク
- T-SQL ステートメント実行タスク
- 式タスク
- ファイルシステムタスク
- オペレーター通知タスク
- メール送信タスク

サポートされているSSIS コンテナは次のとおりです。

- ループコンテナ用
- Foreach ループコンテナ
- シーケンスコンテナ

AWS SCT を使用して SSIS を AWS Glue Studio に変換する

AWS SCT を使用して、から Microsoft SQL Server Integration Services (SSIS) パッケージを AWS Glue Studio に変換できます。

SSIS パッケージには、特定の抽出、変換、ロード (ETL) タスクを実行するために必要なコンポーネント (接続マネージャ、タスク、制御フロー、データフロー、パラメータ、イベントハンドラー、変数など) が含まれています。AWS SCT は、SSIS パッケージを AWS Glue Studio と互換性のある形式に変換します。ソースデータベースを AWS クラウド に移行したら、変換した AWS Glue Studio ジョブを実行して ETL タスクを実行できます。

Microsoft SSIS パッケージを AWS Glue Studio に変換するには、AWS SCT バージョン 1.0.661 以降を使用していることを確認してください。

トピック

前提条件

このセクションでは、SSIS パッケージを AWS Glue に変換するための前提条件となるタスクについて説明します。これらのタスクには、アカウントで必要な AWS リソースを作成することが含まれます。

AWS Identity and Access Management (IAM) を使用して、リソースにアクセスするために AWS Glue Studio が使用する必要なポリシーとロールを定義します。詳細については、「AWS Glue Studio ユーザーの IAM アクセス許可」を参照してください。

AWS SCT がソーススクリプトを AWS Glue Studio に変換したら、変換したスクリプトを Amazon S3 バケットにアップロードします。この Amazon S3 バケットは必ず作成し、AWS サービスプロファイル設定で選択してください。Amazon S3 バケットの作成の詳細については、『Amazon Simple Storage Service ユーザーガイド』の「最初の S3 バケットの作成」を参照してください。

AWS Glue Studio がデータストアに接続できることを確認するには、カスタムコネクタと接続を作成します。また、データベースの認証情報を AWS Secrets Manager に保存してください。

カスタムコネクタを作成するには

1. データストア用の JDBC ドライバーをダウンロードします。AWS SCT が使用する JDBC ドライバーの詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。
2. このドライバーを Amazon S3 バケットにアップロードします。詳細については、『Amazon Simple Storage Service ユーザーガイド』の「バケットへのオブジェクトのアップロード」を参照してください。
3. AWS Management Console にサインインして、AWS Glue Studio コンソール (https://console.aws.amazon.com/gluestudio/) を開きます。
4. [コネクタ] を選択し、[カスタムコネクタの作成] を選択します。
5. [コネクタ S3 URL] で [S3 を参照] を選択し、Amazon S3 バケットにアップロードした JDBC ドライバーファイルを選択します。
6. コネクタのわかりやすい [名前] を入力します。たとえば、**SQLServer** と入力します。
7. [コネクタタイプ] には JDBC を選択します。
8. [クラス名] には、JDBC ドライバーのメインクラスの名前を入力します。SQL Server の場合は、**com.microsoft.sqlserver.jdbc.SQLServerDriver** と入力します。
9. [JDBC URL ベース] の場合は、JDBC ベース URL を入力します。JDBC ベース URL の構文は、ソースデータベースエンジンによって異なります。SQL Server の場合は、**jdbc:sqlserver://$*<host>*:$*<port>*;databaseName=$*<dbname>*;user=$*<username>*;password=$*<password>*** の形式を使用します。

   <host>、*<port>*、*<dbname>*、*<username>*、*<password>* は必ず自分の値に置き換えてください。
10. [URL パラメータの区切り文字] には、セミコロン (;) を入力します。
11. [コネクターを作成] をクリックします。

AWS Secrets Manager にデータベース認証情報を保存するには

1. AWS Management Console にサインインし、AWS Secrets Manager コンソール (https://console.aws.amazon.com/secretsmanager/) を開きます。
2. [Store a new secret] (新しいシークレットの保存) を選択します。
3. [Choose secret type] (シークレットタイプを選択する) ページで、次の操作を行います。

   a. [シークレットタイプ] で、[他の種類のシークレット] を選択します。

   b. [キー/値ペア] () で、**host**、**port**、**dbname**、**username**、**password** のキーを入力します。

   次に、これらのキーの値を入力します。
4. [シークレットの設定] ページで、わかりやすい [シークレット名] を入力します。たとえば、**SQL_Server_secret** と入力します。
5. [次へ] をクリックします。[自動回転の設定] ページで、[次へ] を選択します。
6. [Review] (レビュー) ページで、シークレットの詳細を確認し、[Store] (保存) を選択します。

コネクタ用に接続を作成するには

1. AWS Management Console にサインインして、AWS Glue Studio コンソール (https://console.aws.amazon.com/gluestudio/) を開きます。
2. 接続を作成するコネクタを選択し、[接続の作成] をクリックします。
3. [接続の作成] ページで、接続のわかりやすい [名前]、およびその説明を入力します。たとえば、**SQL-Server-connection** と入力します。
4. [AWS シークレット] には、AWS Secrets Manager で作成したシークレットを選択します。
5. [ネットワークオプション] を設定し、[接続の作成] を選択します。

これで、カスタムコネクタを使用して AWS Glue Studio ジョブを作成できます。詳細については、「AWS Glue Studio ジョブの作成」を参照してください。

SSIS パッケージを AWS SCT プロジェクトに追加する

1 つの AWS SCT プロジェクトに複数の SSIS パッケージを追加できます。

SSIS パッケージを AWS SCT プロジェクトに追加するには

1. AWS SCT で新しいプロジェクトを作成するか、既存のプロジェクトを開きます。詳細については、「the section called “プロジェクトの作成”」を参照してください。
2. メニューから [ソースを追加] を選択し、[SQL Server 統合サービス] を選択します。
3. [接続名] では、SSIS パッケージの名前を入力します。 AWS SCT の左パネルのツリーにこの名前が表示されます。
4. [SSIS パッケージフォルダ] には、ソース SSIS パッケージが保存されているフォルダへのパスを入力します。
5. メニューから、[ターゲットの追加]、[AWS Glue Studio] の順に選択します。

   AWS Glue Studio に接続するために、AWS SCT は AWS プロファイルを使用します。詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。
6. ソース SSIS パッケージと AWS Glue Studio ターゲットを含むマッピングルールを作成します。詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。
7. AWS Glue Studioコンソールで AWS Glue Studio 接続を作成します。詳細については、「コネクタ用の接続を作成する」を参照してください。
8. 左側のツリーで [接続マネージャ] コンテキスト (右クリック) メニューを開き、[接続の設定] を選択します。

   AWS SCT で、[接続の設定] ウィンドウが表示されます。
9. ソース SSIS 接続ごとに、AWS Glue Studio 接続を選択します。

AWS SCT を使用して SSIS パッケージを AWS Glue Studio に変換する

次に、AWS SCT を使用して SSIS パッケージを AWS Glue Studio に変換する方法を示します。

SSIS パッケージを AWS Glue Studio に変換するには

1. SSIS パッケージを AWS SCT プロジェクトに追加します。詳細については、「SSIS パッケージを AWS SCT プロジェクトに追加する」を参照してください。
2. 左側のパネルで [ETL] ノードと [SSIS] ノードを展開します。
3. [パッケージ] を選択し、コンテキスト (右クリック) メニューを開いてから、[パッケージの変換] を選択します。

AWS SCT は選択した SSIS パッケージを JSON ファイルに変換します。これらの JSON オブジェクトは、有向非巡回グラフ (DAG) 内のノードを表します。右側のツリーの [パッケージ DAG] ノードで、変換したファイルを見つけます。

4. コンテキスト (右クリック) メニューから [パッケージ DAG] を開いてから、[S3 に保存する] を選択します。

   これで、これらのスクリプトを使用して AWS Glue Studio でジョブを作成できます。

変換したコードを使用して AWS Glue Studio ジョブを作成する

ソース SSIS パッケージを変換したら、変換された JSON ファイルを使用して AWS Glue Studio ジョブを作成できます。

AWS Glue Studio ジョブを作成するには

1. コンテキスト (右クリック) メニューから [パッケージ DAG] を選択してから、[CAWS Glue Studio ジョブの設定] を選択します。
2. (オプション) SSIS 関数を AWS Glue Studio でエミュレートする拡張パックを適用します。
3. [AWS Glue Studio ジョブの設定] ウィンドウが開きます。

   [基本ジョブのプロパティ] セクションに必要事項を入力します。

   - [名前] — AWS Glue Studio ジョブの名前を入力します。
   - [スクリプトファイル名] — ジョブスクリプトの名前を入力します。
   - [パラメータの追加] を選択し、その値を入力します。

   [次へ] をクリックします。
4. [アドバンスジョブのプロパティ] セクションに必要事項を入力します。

   - [IAM ロール] — AWS Glue Studio データストアへの認証とアクセスに使用する IAM ロールを選択します。
   - [スクリプトファイル S3 パス] — 変換されたスクリプトへの Amazon S3 パスを入力します。
   - [一時ディレクトリ] — 中間結果用の一時ディレクトリへの Amazon S3 パスを入力します。AWS Glue Studio はこのディレクトリを使用して、Amazon Redshift への読み取りまたは書き込みを行います。

- AWS SCT は Python ライブラリのパスを自動的に生成します。このパスは [生成された Python ライブラリパス] で確認できます。この自動生成されたパスは編集できません。追加の Python ライブラリを使用するには、[ユーザ Python ライブラリパス] にパスを入力します。
- [ユーザ Python ライブラリパス] — 追加のユーザ Python ライブラリのパスを入力します。Amazon S3 のパスはカンマで区切ります。
- [依存 jars パス] – 依存 `*.jar` ファイルのパスを入力します。Amazon S3 のパスはカンマで区切ります。
- [参照ファイルパス] — 設定ファイルなど、スクリプトに必要な追加ファイルのパスを入力します。Amazon S3 のパスはカンマで区切ります。
- [ワーカータイプ] — `G.1X` または `G.2X` を選択します。

  `G.1X` を選択すると、各ワーカーは 1 DPU (4 vCPU、16 GB のメモリ、64 GB のディスク) にマッピングされます。

  `G.2X` を選択すると、各ワーカーは 2 DPU (8 vCPU、32 GB のメモリ、128 GB のディスク) にマッピングされます。
- [要求されたワーカー数] — ジョブの実行時に割り当てられるワーカーの数を入力します。
- [同時実行の最大数] – このジョブで許可される同時実行の最大数を設定します。デフォルトは 1 です。このしきい値に達すると、AWS Glue はエラーを返します。
- [ジョブタイムアウト (分)] — ジョブの暴走を防ぐため、ETL ジョブのタイムアウト値を入力します。バッチジョブのデフォルト値は 2,880 分 (48 時間) です。ジョブがこの制限を超えると、ジョブの実行状態は `TIMEOUT` に変わります。
- [遅延通知のしきい値 (分)] — AWS SCT 遅延通知を送信する前のしきい値を分単位で入力します。
- [リトライ回数] — AWS Glue 失敗した場合にジョブを自動的に再開する回数 (0 ~ 10 回) を入力します。タイムアウト制限に達したジョブは再起動されません。デフォルトは 0 です。

[終了] を選択します。

AWS SCT は選択した AWS Glue Studio ジョブを設定します。

5. 右側のツリーの [ETL ジョブ] で、設定したジョブを探します。設定したジョブを選択し、コンテキスト (右クリック) メニューを開いてから、[AWS Glue Studio ジョブの作成] を選択します。
6. [適用ステータス] を選択し、ジョブの [ステータス] の値が [成功] であることを確認します。

7. AWS Glue Studio コンソールを開き、[更新] を選択し、ジョブを選択します。その後、[Run] (実行) をクリックします。

AWS SCT を使用して SSIS パッケージの評価レポートを作成します。

ETL 移行評価レポートには、AWS Glue Studio を使用して SSIS パッケージをと互換性のある形式に変換する方法に関する情報が記載されています。評価レポートには、SSIS パッケージのコンポーネントに関するアクション項目が含まれています。これらのアクション項目には、AWS SCT が自動的に変換できないコンポーネントが示されます。

ETL 移行評価レポートを作成するには

1. 左側のパネルの [ETL] の下にある [SSIS] ノードを展開します。
2. [パッケージ] を選択して オブジェクトのコンテキスト (右クリック) メニューを開き、[レポートを作成] を選択します。
3. [概要] タブを表示します。ここには、AWS SCT によって ETL 移行評価レポートのエグゼクティブサマリー情報が表示されます。SSIS パッケージのすべてのコンポーネントの変換結果が含まれます。
4. (オプション) ETL 移行評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらでも保存できます。

   - ETL 移行評価レポートを PDF ファイルとして保存するには、右上の [PDF に保存] を選択します。

     PDF ファイルには、スクリプト変換に関するエグゼクティブサマリー、アクション項目、推奨事項が含まれています。

   - ETL 移行評価レポートを CSV ファイルとして保存するには、右上の [CSV に保存] を選択します。

     AWS SCT は 3 つの CSV ファイルを作成します。これらのファイルには、アクション項目、推奨アクション、およびスクリプトの変換に必要な推定手作業の複雑さが含まれています。

5. [アクション項目] タブを選択します。このタブには、AWS Glue Studio への手動変換が必要な項目のリストが含まれます。リストからアクション項目を選択すると、AWS SCT はそのアクション項目が適用されるソース SSIS パッケージの項目を強調表示します。

AWS SCT で AWS Glue Studio に変換できる SSIS コンポーネント

AWS SCTを使用して SSIS データフローコンポーネントとパラメータを AWS Glue Studio に変換できます。

サポートされるデータフローコンポーネントは次のとおりです。

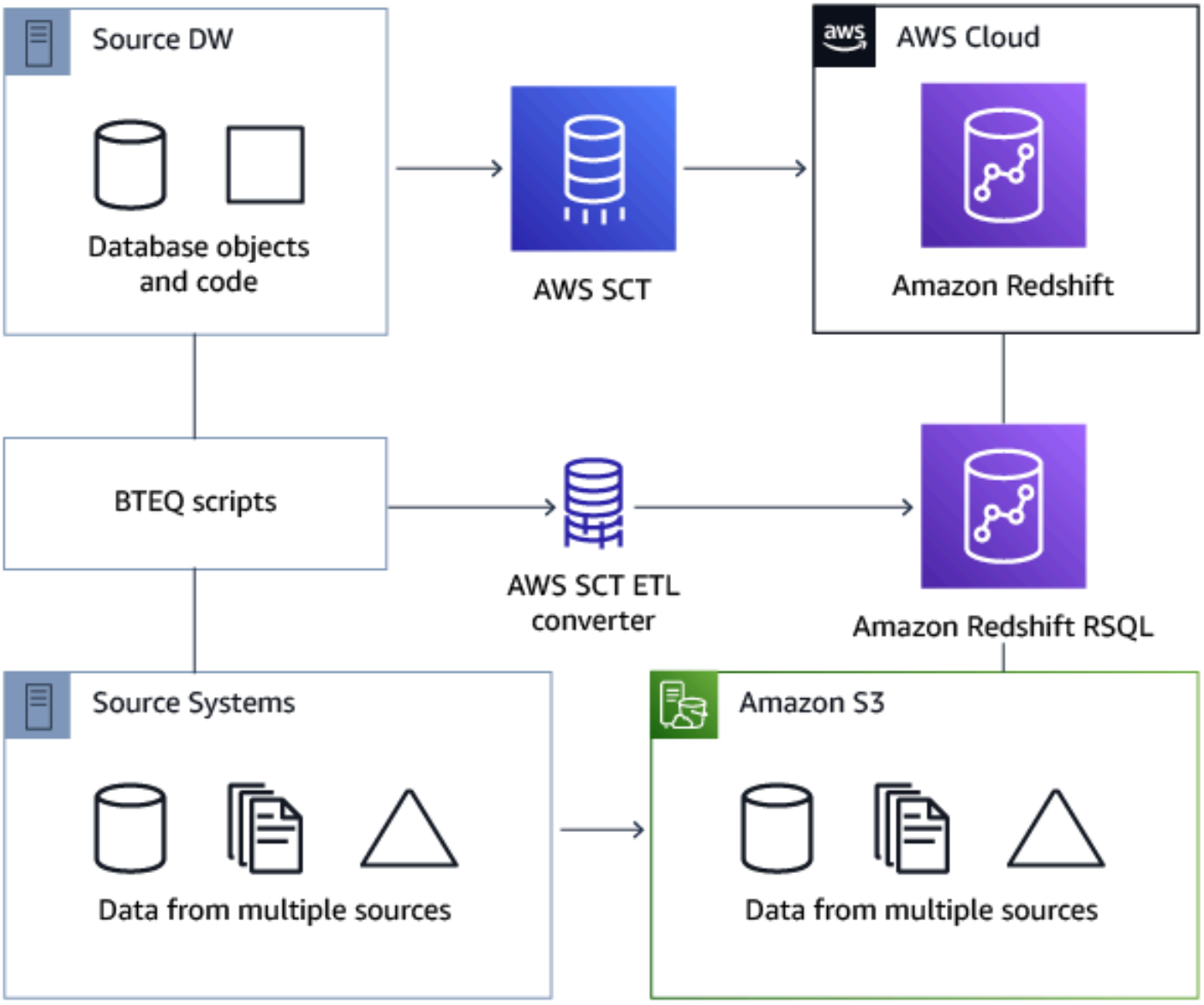
- ADO NET 送信先
- ADO NET 送信元
- 集計
- 文字マップ
- 条件付き分割
- コピー列
- データ変換
- 派生カラム
- 検索
- マージ
- マージ結合
- マルチキャスト
- ODBC 送信先
- ODBC 送信元
- OLEDB 送信先
- OLEDB 送信元
- 行数
- 並べ替え
- SQL Server 送信先
- Union All

AWS SCT はより多くの SSIS コンポーネントを AWS Glue に変換できます。詳細については、「AWS SCT が AWS Glue に変換できる SSIS コンポーネント」を参照してください。

AWS SCT を使用してテラデータ BTEQ スクリプトを Amazon Redshift RSQL に変換する

AWS Schema Conversion Tool(AWS SCT) を使用して、Teradata Basic Teradata Query (BTEQ) スクリプトを Amazon Redshift RSQL に変換できます。

次のアーキテクチャ図は、抽出、変換、ロード (ETL) スクリプトから Amazon Redshift RSQL への変換を含むデータベース移行プロジェクトを示しています。



トピック

- BTEQ スクリプトを AWS SCT プロジェクトに追加する

- BTEQ スクリプトの代替変数を AWS SCT で設定する
- AWS SCT を使用してテラデータ BTEQ スクリプトを Amazon Redshift RSQL に変換する
- AWS SCT を使用した BTEQ スクリプトの管理
- AWS SCT での BTEQ スクリプト変換評価レポートの作成
- AWS SCT を使用して変換された BTEQ スクリプトの編集と保存

BTEQ スクリプトを AWS SCT プロジェクトに追加する

複数のスクリプトを単一の AWS SCT プロジェクトに追加できます。

BTEQ スクリプトを AWS SCT プロジェクトに追加するには

1. AWS SCT で新しいプロジェクトを作成するか、既存のプロジェクトを開きます。詳細については、「the section called “プロジェクトの作成”」を参照してください。
2. メニューから [ソースを追加] を選択し、次に [Teradata] を選択してソースデータベースをプロジェクトに追加します。詳細については、「ソースとしての Teradata の使用」を参照してください。
3. メニューから [ターゲットを追加] を選択し、ターゲット Amazon Redshift データベースを AWS SCT プロジェクトに追加します。

   仮想 Amazon Redshift ターゲットデータベースプラットフォームを使用できます。詳細については、「仮想ターゲットの使用」を参照してください。
4. ソース Teradata データベースと Amazon Redshift ターゲットを含む新しいマッピングルールを作成します。詳細については、「新しいマッピングルールの追加」を参照してください。
5. [ビュー] メニューで、[メインビュー] を選択します。
6. 左側のパネルで、[スクリプト] ノードを展開します。
7. [BTEQ スクリプト] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトのロード] を選択します。
8. Teradata BTEQ スクリプトのソースコードの場所を入力し、[フォルダの選択] を選択します。

   AWS SCT で、[スクリプトのロード] ウィンドウが表示されます。
9. 次のいずれかを実行します。

   a. Teradata BTEQ スクリプトに代替変数が含まれていない場合は、[代替変数なし] を選択し、[OK] を選択してスクリプトを AWS SCT プロジェクトに追加します。

b. Teradata BTEQ スクリプトに代替変数が含まれている場合は、代替変数を設定します。詳細については、「BTEQ スクリプトでの代替変数の設定」を参照してください。

BTEQ スクリプトの代替変数を AWS SCT で設定する

Teradata BTEQ スクリプトには代替変数を含めることができます。たとえば、1 つの BTEQ スクリプトと代替変数を使用して、複数のデータベース環境で同じコマンドセットを実行できます。AWS SCT を使用して BTEQ スクリプト内の代替変数を設定できます。

代替変数を使用して BTEQ スクリプトを実行する前に、必ずすべての変数に値を割り当ててください。そのためには、Bash スクリプト、UC4 (Automic) など、他のツールやアプリケーションを使用できます。AWS SCT は、値を割り当てた後でのみ置換変数を解決および変換できます。

BTEQ スクリプトで代替変数を設定するには

1. BTEQ スクリプトを AWS SCT プロジェクトに追加します。詳細については、「BTEQ スクリプトを AWS SCT プロジェクトに追加する」を参照してください。

   スクリプトを追加するときは、[代替変数が使用される] を選択します。

2. [変数フォーマットの定義] には、スクリプト内のすべての代替変数と一致する正規表現を入力します。

   たとえば、代替変数の名前が ${ で始まり、} で終わる場合は、`\$\{\w+\}` 正規表現を使用します。ドル記号またはパーセント記号で始まる代替変数を照合するには、`\$\w+|\%\w+` 正規表現を使用します。

   AWS SCT の正規表現は、Java の正規表現の構文に準拠しています。詳細については、Java ドキュメントの「java.util.regex クラスパターン」を参照してください。

3. [OK] を選択してスクリプトを AWS SCT プロジェクトにロードし、[OK] を選択して [スクリプトのロード] ウィンドウを閉じます。

4. [変数] を選択すると、検出されたすべての代替変数とその値が表示されます。

5. [値] には、代替変数の値を入力します。

AWS SCT を使用してテラデータ BTEQ スクリプトを Amazon Redshift RSQL に変換する

以下では、AWS SCT を使用して BTEQ ETL スクリプトを Amazon Redshift RSQL に変換する方法について説明します。

テラデータ BTEQ スクリプトを Amazon Redshift RSQL に変換するには

1. BTEQ スクリプトを AWS SCT プロジェクトに追加します。詳細については、「BTEQ スクリプトを AWS SCT プロジェクトに追加する」を参照してください。
2. 代替変数を設定します。詳細については、「BTEQ スクリプトでの代替変数の設定」を参照してください。
3. 左側のパネルで、[スクリプト] ノードを展開します。
4. 次のいずれかを実行します。

   - 1 つの BTEQ スクリプトを変換するには、[BTEQスクリプト] ノードを展開し、変換するスクリプトを選択して、コンテキスト (右クリック) メニューから [Convert to RSQL に変換] を選択します。
   - 複数のスクリプトを変換するには、変換するスクリプトをすべて選択してください。次に [BTEQ スクリプト] を選択し、コンテキスト (右クリック) メニューを開いて、[スクリプトの変換] で [RSQL に変換] を選択します。

   AWS SCT は、選択したすべての Teradata BTEQ スクリプトを Amazon Redshift RSQL と互換性のある形式に変換します。変換されたスクリプトは、ターゲットデータベースパネルの [スクリプト] ノードにあります。
5. 変換した Amazon Redshift RSQL スクリプトを編集するか、保存してください。詳細については、「変換された BTEQ スクリプトの編集と保存」を参照してください。

AWS SCT を使用した BTEQ スクリプトの管理

複数の BTEQ スクリプトを追加したり、AWS SCT プロジェクトから BTEQ スクリプトを削除したりできます。

BTEQ スクリプトを AWS SCT プロジェクトに追加するには

1. 左パネルの [スクリプト] ノードを展開します。

2. [BTEQ スクリプト] ノードを開き、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトのロード] を選択します。
4. 新しい BTEQ スクリプトを追加し、代替変数を設定するのに必要な情報を入力します。詳細については、「BTEQ スクリプトを AWS SCT プロジェクトに追加する」および「BTEQ スクリプトでの代替変数の設定」を参照してください。

BTEQ スクリプトを AWS SCT プロジェクトから削除するには

1. 左パネルの[スクリプト] の下にある [BTEQ スクリプト] ノードを展開します。
2. 削除するスクリプトを選択して、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトを削除] を選択します。

AWS SCT での BTEQ スクリプト変換評価レポートの作成

BTEQ スクリプト変換評価レポートには、BTEQ スクリプトの BTEQ コマンドと SQL ステートメントを Amazon Redshift RSQL と互換性のある形式に変換する方法に関する情報が記載されています。評価レポートには、BTEQ コマンドのアクション項目と、AWS SCT が変換できない SQL ステートメントが含まれます。

BTEQ スクリプト変換評価レポートを作成するには

1. 左側のパネルの [スクリプト] にある [BTEQ スクリプト] ノードを展開します。
2. 変換するスクリプトを選択して、オブジェクトのコンテキスト (右クリック) メニューを開きます。
3. [レポートの作成] で [RSQL に変換] を選択します。
4. [概要] タブを表示します。[概要] タブには、BTEQ スクリプト評価レポートの要約が表示されます。BTEQ スクリプトからのすべての BTEQ コマンドと SQL ステートメントの変換結果が含まれます。
5. (オプション) BTEQ スクリプト変換評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらかとして保存します。

   - BTEQ スクリプト変換評価レポートを PDF ファイルとして保存するには、右上の [PDF に保存] を選択します。

     PDF ファイルには、スクリプト変換に関するエグゼクティブサマリー、アクション項目、推奨事項が含まれています。

- BTEQ スクリプト変換評価レポートを CSV ファイルとして保存するには、右上の [Save to CSV] を選択します。

  CSV ファイルには、アクション項目、推奨アクション、およびスクリプトの変換に必要な推定手作業の複雑さが含まれています。

6. [アクション項目] タブを選択します。このタブには、Amazon Redshift RSQL への手動変換が必要な項目のリストが含まれます。リストからアクション項目を選択すると、そのアクション項目が適用されるソース BTEQ スクリプトの項目が AWS SCT によって強調表示されます。

AWS SCT を使用して変換された BTEQ スクリプトの編集と保存

変換したスクリプトは AWS SCT プロジェクトの下部パネルで編集できます。AWS SCT は、編集された、プロジェクトの一部として編集された、スクリプトを保存します。

変換されたスクリプトを保存するには

1. ターゲットデータベースパネルの [スクリプト] にある [RSQL スクリプト] ノードを展開します。
2. 変換された、コンテキスト (右クリック) メニューを開いてから、[スクリプトを保存] を選択します。
3. 変換したスクリプトを保存するフォルダへのパスを入力し、[保存] を選択します。

   AWS SCT は変換したスクリプトをファイルに保存し、このファイルを開きます。

AWS SCT を使用して、Teradata BTEQ コマンドが埋め込まれているシェルスクリプトを Amazon Redshift RSQL に変換する

AWS Schema Conversion Tool(AWS SCT) を使用して、Teradata ベーシックテラデータクエリ (BTEQ) コマンドが埋め込まれているシェルスクリプトを、Amazon Redshift RSQL コマンドが埋め込まれているシェルスクリプトに変換できます。

AWS SCT はシェルスクリプトから Teradata BTEQ コマンドを抽出し、Amazon Redshift と互換性のある形式に変換します。Teradata データベースを Amazon Redshift に移行すると、変換されたこれらのスクリプトを使用して新しい Amazon Redshift データベースを管理できます。

AWS SCT を使用して、Teradata BTEQ ETL スクリプトを含むファイルを Amazon Redshift RSQL に変換することもできます。詳細については、「AWS SCT を使用してテラデータ BTEQ スクリプトを Amazon Redshift RSQL に変換する」を参照してください。

トピック

- Teradata BTEQ コマンドが埋め込まれたシェルスクリプトを AWS SCT プロジェクトに追加する
- AWS SCT を使用した埋め込み Teradata BTEQ コマンドを使用したシェルスクリプトでの置換変数の設定
- AWS SCT を使用した、Teradata BTEQ コマンドが埋め込まれているシェルスクリプトの変換
- AWS SCT を使用した、Teradata BTEQ コマンドが組み込まれたシェルスクリプトの管理
- AWS SCT を使用してシェルスクリプト変換用の評価レポートを作成します。
- AWS SCT で変換された、シェルスクリプトの編集と保存

Teradata BTEQ コマンドが埋め込まれたシェルスクリプトを AWS SCT プロジェクトに追加する

複数のスクリプトを単一の AWS SCT プロジェクトに追加できます。

シェルスクリプトを AWS SCT プロジェクトに追加するには

1. AWS SCT で新しいプロジェクトを作成するか、既存のプロジェクトを開きます。詳細については、「the section called “プロジェクトの作成”」を参照してください。
2. メニューから [ソースを追加] を選択し、次に [Teradata] を選択してソースデータベースをプロジェクトに追加します。詳細については、「ソースとしての Teradata の使用」を参照してください。
3. メニューから [ターゲットを追加] を選択し、ターゲットの Amazon Redshift AWS SCT データベースをプロジェクトに追加します。

   仮想 Amazon Redshift ターゲットデータベースプラットフォームを使用できます。詳細については、「仮想ターゲットの使用」を参照してください。
4. ソース Teradata データベースと Amazon Redshift ターゲットを含む新しいマッピングルールを作成します。詳細については、「新しいマッピングルールの追加」を参照してください。
5. [ビュー] メニューで、[メインビュー] を選択します。
6. 左側のパネルで、[スクリプト] ノードを展開します。

7. [シェル] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトのロード] を選択します。
8. Teradata BTEQ コマンドが埋め込まれているソースシェルスクリプトの場所を入力し、[フォルダの選択] を選択します。

   AWS SCT[スクリプトのロード] ウィンドウが表示されます。
9. 次のいずれかを実行します。

   - シェルスクリプトに代替変数が含まれていない場合は、[代替変数なし] を選択し、[OK] を選択してスクリプトを AWS SCT プロジェクトに追加します。
   - シェルスクリプトに代替変数が含まれている場合は、代替変数を設定します。詳細については、「シェルスクリプトでの代替変数の設定」を参照してください。

AWS SCT を使用した埋め込み Teradata BTEQ コマンドを使用したシェルスクリプトでの置換変数の設定

シェルスクリプトには代替変数を含めることができます。たとえば、代替変数を含む単一のスクリプトを使用して、さまざまな環境のデータベースを管理できます。AWS SCT を使用して、シェルスクリプトで代替変数を設定できます。

シェルスクリプトから代替変数を使用して BTEQ コマンドを実行する前に、必ずこのシェルスクリプト内のすべての変数に値を割り当ててください。 AWS SCT が代替変数を解決および変換できるのは、値を割り当てた後のみです。

シェルスクリプトで代替変数を設定するには

1. AWS SCTソースシェルスクリプトをプロジェクトに追加します。詳細については、「シェルスクリプトを AWS SCT プロジェクトに追加する」を参照してください。

   スクリプトを追加するときは、[代替変数が使用される] を選択します。
2. [変数フォーマットの定義] には、スクリプト内のすべての代替変数と一致する正規表現を入力します。

   たとえば、代替変数の名前が `${` で始まり、`}` で終わる場合は、`\$\{\w+\}` 正規表現を使用します。ドル記号またはパーセント記号で始まる代替変数を照合するには、`\$\w+|\%\w+` 正規表現を使用します。

AWS SCT の正規表現は、Java の正規表現の構文に準拠しています。詳細については、Java ドキュメントの「java.util.regex クラスパターン」を参照してください。

3. [OK] を選択してスクリプトを AWS SCT プロジェクトにロードし、[OK] を選択して [スクリプトのロード] ウィンドウを閉じます。
4. [変数] を選択すると、検出されたすべての代替変数とその値が表示されます。
5. [値] には、代替変数の値を入力します。

AWS SCT を使用した、Teradata BTEQ コマンドが埋め込まれているシェルスクリプトの変換

以下では、AWS SCT を使用して Teradata BTEQ コマンドが埋め込まれているシェルスクリプトを、Amazon Redshift RSQL コマンドが埋め込まれているシェルスクリプトに変換する方法について説明します。

シェルスクリプトを変換するには

1. シェルスクリプトを AWS SCT プロジェクトに追加します。詳細については、「シェルスクリプトを AWS SCT プロジェクトに追加する」を参照してください。
2. 代替変数を設定します。詳細については、「シェルスクリプトでの代替変数の設定」を参照してください。
3. 左側のパネルで、[スクリプト] ノードを展開します。
4. 次のいずれかを実行します。

   - 1 つのシェルスクリプトから BTEQ コマンドを変換するには、[シェル] ノードを展開し、変換するスクリプトを選択し、コンテキスト (右クリック) メニューから [スクリプトを変換] を選択します。
   - 複数のスクリプトを変換するには、変換するスクリプトをすべて選択してください。[シェル] を選択してコンテキスト (右クリック) メニューを開き、[スクリプトを変換] を選択します。
5. [OK] をクリックします。

   AWS SCT は、選択したシェルスクリプト内の BTEQ コマンドを Amazon Redshift RSQL と互換性のある形式に変換します。変換されたスクリプトは、ターゲットデータベースパネルの [スクリプト] ノードにあります。
6. 変換した Amazon Redshift RSQL スクリプトを編集するか、保存してください。詳細については、「変換されたシェルスクリプトの編集と保存」を参照してください。

AWS SCT を使用した、Teradata BTEQ コマンドが組み込まれたシェルスクリプトの管理

複数のシェルスクリプトを追加したり、AWS SCT プロジェクトからシェルスクリプトを削除したりできます。

新しいシェルスクリプトを AWS SCT プロジェクトに追加するには

1. 左パネルの [スクリプト] ノードを展開します。
2. [シェル] ノードを選択して、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトのロード] を選択します。
4. 新しいシェルスクリプトを追加し、代替変数を設定するのに必要な情報を入力します。詳細については、「シェルスクリプトを AWS SCT プロジェクトに追加する」および「シェルスクリプトでの代替変数の設定」を参照してください。

AWS SCT プロジェクトからシェルスクリプトを削除するには

1. 左パネルの [スクリプト] の下の [シェル] ノードを展開します。
2. 削除するスクリプトを選択して、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトを削除] を選択します。

AWS SCT を使用してシェルスクリプト変換用の評価レポートを作成します。

シェルスクリプト変換評価レポートには、BTEQ コマンドと SQL ステートメントの変換に関する情報が記載されています。ソーススクリプトから Amazon Redshift RSQL と互換性のある形式への変換が行われます。評価レポートには、BTEQ コマンドのアクション項目と、AWS SCT で変換できない SQL ステートメントが含まれています。

シェルスクリプト変換評価レポートを作成するには

1. 左パネルの [スクリプト] の下の [シェル] ノードを展開します。
2. 変換するスクリプトを選択し、コンテキスト (右クリック) メニューを開いてから、[レポートの作成] を選択します。

3. [概要] タブを表示します。[概要] タブには、シェルスクリプト評価レポートのエグゼクティブサマリーが表示されます。これには、ソーススクリプトからのすべての BTEQ コマンドと SQL ステートメントの変換結果が含まれます。
4. (オプション) シェルスクリプト変換評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらかとして保存します。

   - シェルスクリプト変換評価レポートを PDF ファイルとして保存するには、右上の [PDF 二保存] を選択します。

     PDF ファイルには、スクリプト変換に関するエグゼクティブサマリー、アクション項目、推奨事項が含まれています。
   - シェルスクリプト変換評価レポートを CSV ファイルとして保存するには、右上の [CSV に保存] を選択します。

     CSV ファイルには、アクション項目、推奨アクション、およびスクリプトの変換に必要な推定手作業の複雑さが含まれています。
5. [アクション項目] タブを選択します。このタブには、Amazon Redshift RSQL への手動変換が必要な項目のリストが含まれます。リストからアクション項目を選択すると、アクション項目が適用されるシェルスクリプトから、その項目が AWS SCT によって強調表示されます。

AWS SCT で変換された、シェルスクリプトの編集と保存

変換したスクリプトは AWS SCT プロジェクトの下部パネルで編集できます。AWS SCT は、プロジェクトの一部として編集されたスクリプトを保存します。

変換されたスクリプトを保存するには

1. ターゲットデータベースパネルの [RSQL スクリプト] にある [スクリプト] ノードを展開します。
2. 変換された、コンテキスト (右クリック) メニューを開いて、[スクリプトの保存] を選択します。
3. 変換したスクリプトを保存するフォルダへのパスを入力し、[保存] を選択します。

   AWS SCT は変換したスクリプトをファイルに保存し、このファイルを開きます。

AWS SCT を使用してた Teradata FastExport ジョブスクリプトの Amazon Redshift RSQL への変換

AWS Schema Conversion Tool(AWS SCT) を使用して Teradata FastExport ジョブスクリプトを Amazon Redshift RSQL に変換できます。

FastExport ジョブスクリプトは、Teradata データベースからデータを選択してエクスポートする FastExport コマンドと SQL ステートメントのセットです。AWS SCT は、FastExport コマンドと SQL ステートメントを Amazon Redshift RSQL と互換性のある形式に変換します。Teradata データベースを Amazon Redshift に移行したら、変換されたこれらのスクリプトを使用して Amazon Redshift データベースからデータをエクスポートできます。

トピック

FastExport ジョブスクリプトを AWS SCT プロジェクトに追加する

複数のスクリプトを単一の AWS SCT プロジェクトに追加できます。

FastExport ジョブスクリプトを AWS SCT プロジェクトに追加するには

1. AWS SCT で新しいプロジェクトを作成するか、既存のプロジェクトを開きます。詳細については、「the section called “プロジェクトの作成”」を参照してください。
2. メニューから [ソースを追加] を選択し、次に [Teradata] を選択してソースデータベースをプロジェクトに追加します。詳細については、「ソースとしての Teradata の使用」を参照してください。
3. メニューから [ターゲットを追加] を選択し、ターゲットの Amazon Redshift AWS SCT データベースをプロジェクトに追加します。

   仮想 Amazon Redshift ターゲットデータベースプラットフォームを使用できます。詳細については、「仮想ターゲットの使用」を参照してください。

4. ソース Teradata データベースと Amazon Redshift ターゲットを含む新しいマッピングルールを作成します。詳細については、「新しいマッピングルールの追加」を参照してください。
5. [ビュー] メニューで、[メインビュー] を選択します。
6. 左側のパネルで、[スクリプト] ノードを展開します。
7. [FastExport] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトのロード] を選択します。
8. Teradata FastExport ジョブスクリプトのソースコードの場所を入力し、[フォルダの選択] を選択します。

   AWS SCT で [スクリプトのロード] ウィンドウが表示されます。
9. 次のいずれかを実行します。

   - Teradata FastExport ジョブスクリプトに代替変数が含まれていない場合は、[代替変数なし] を選択し、[OK] を選択してスクリプトを AWS SCT プロジェクトに追加します。
   - Teradata FastExport ジョブスクリプトに代替変数が含まれている場合は、代替変数を設定します。詳細については、「FastExport ジョブスクリプトでの代替変数の設定」を参照してください。

Teradata FastExport ジョブスクリプトの代替変数は、AWS SCT を使用して設定します。

Teradata FastExport ジョブスクリプトには代替変数を含めることができます。たとえば、1 つのスクリプトと代替変数を使用して、複数のデータベースからデータを エクスポートできます。AWS SCT を使用して Teradata スクリプトで代替変数を設定できます。

代替変数を使用して FastExport ジョブスクリプトを実行する前に、必ずすべての変数に値を割り当ててください。そのためには、Bash スクリプト、UC4 (Automic) などの他のツールやアプリケーションを使用できます。AWS SCT は、値を割り当てた後にのみ代替変数を解決および変換できます。

FastExport ジョブスクリプトで代替変数を設定するには

1. Teradata FastExport のソースジョブスクリプトを AWS SCT プロジェクトに追加します。詳細については、「BTEQ スクリプトを AWS SCT プロジェクトに追加する」を参照してください。

   スクリプトを追加するときは、[代替変数が使用される] を選択します。

2. [変数フォーマットの定義] には、スクリプト内のすべての代替変数と一致する正規表現を入力します。

   たとえば、代替変数の名前が ${ で始まり、} で終わる場合は、\$\{\w+\} 正規表現を使用します。ドル記号またはパーセント記号で始まる代替変数を照合するには、\$\w+|\%\w+ 正規表現を使用します。

   AWS SCT の正規表現は、Java の正規表現の構文に準拠しています。詳細については、Java ドキュメントの「java.util.regex クラスパターン」を参照してください。
3. [OK] を選択してスクリプトを AWS SCT プロジェクトにロードし、[OK] を選択して [スクリプトのロード] ウィンドウを閉じます。
4. 左側のパネルで、[スクリプト] ノードを展開します。[FastExport] を選択し、スクリプトが保存されているフォルダを選択します。コンテキスト (右クリック) メニューを開き、[代替変数] で、[変数をエクスポート] を選択します。
5. 1 つのスクリプトの代替変数をエクスポートします。スクリプトを含むフォルダを展開し、スクリプトを選択し、コンテキスト (右クリック) メニューを開き、[代替変数] で [変数をエクスポート] を選択します。
6. 代替変数を保存するカンマ区切り値 (CSV) ファイルの名前を入力し、[保存] を選択します。
7. この CSV ファイルを開き、代替変数の値を入力します。

   オペレーティングシステムによって、AWS SCT は異なる形式の CSV ファイルを使用します。ファイル内の値は、引用符で囲まれていてもいなくてもかまいません。代替変数の値には、ファイル内の他の値と同じ形式を使用してください。AWS SCT は、値が異なる形式の CSV ファイルはインポートできません。
8. CSV ファイルを保存します。
9. 左側のパネルで、[スクリプト] ノードを展開します。[FastExport] を選択し、スクリプトを選択します。コンテキスト (右クリック) メニューを開き、[代替変数] で、[変数をインポート] を選択します。
10. CSV ファイルを選択し、[開く] を選択します。
11. [変数] を選択すると、検出されたすべての代替変数とその値が表示されます。

AWS SCT を使用した Teradata FastExport ジョブスクリプトの変換

以下では、AWS SCT を使用して Teradata FastExport ジョブを Amazon Redshift RSQL に変換する方法について説明します。

Teradata FastExport ジョブスクリプトを Amazon Redshift RSQL に変換するには

1. FastExport ジョブスクリプトを AWS SCT プロジェクトに追加します。詳細については、「FastExport ジョブスクリプトを AWS SCT プロジェクトに追加する」を参照してください。
2. 代替変数を設定します。詳細については、「FastExport ジョブスクリプトでの代替変数の設定」を参照してください。
3. 左側のパネルで、[スクリプト] ノードを展開します。
4. 次のいずれかを実行します。

   - 1 つの FastExport ジョブスクリプトを変換するには、[FastExport] ノードを展開し、変換するスクリプトを選択して、コンテキスト (右クリック) メニューから [スクリプトを変換] を選択します。
   - 複数のスクリプトを変換するには、変換するスクリプトをすべて選択してください。次に、[FastExport] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトを変換] を選択します。

   AWS SCT は、選択したすべての Teradata FastExport ジョブスクリプトを Amazon Redshift RSQL と互換性のある形式に変換します。変換されたスクリプトは、ターゲットデータベースパネルの [スクリプト] ノードにあります。
5. 変換した Amazon Redshift RSQL スクリプトを編集するか、保存してください。詳細については、「変換された FastExport ジョブスクリプトの編集と保存」を参照してください。

AWS SCT を使用した Teradata FastExport ジョブスクリプトの管理

複数の Teradata FastExport ジョブスクリプトを追加したり、FastExport ジョブスクリプトを AWS SCT プロジェクトから削除したりできます。

新しい FastExport ジョブスクリプトを AWS SCT プロジェクトに追加するには

1. 左パネルの [スクリプト] ノードを展開します。
2. [FastExport] ノードを選択し、ファイルのコンテキストメニュー (右クリック) を開きます。
3. [スクリプトのロード] を選択します。
4. 新しい FastExport ジョブスクリプトを追加し、代替変数を設定するのに必要な情報を入力します。詳細については、「FastExport ジョブスクリプトを AWS SCT プロジェクトに追加する」および「FastExport ジョブスクリプトでの代替変数の設定」を参照してください。

FastExport ジョブスクリプトを AWS SCT プロジェクトから削除するには

1. 左側のパネルの [スクリプト] の下にある [FastExport] ノードを展開します。
2. 削除するスクリプトを選択して、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトを削除] を選択します。

AWS SCT を使用した Teradata FastExport ジョブスクリプト変換の評価レポートの作成

FastExport ジョブスクリプト変換評価レポートには、FastExport スクリプトの FastExport コマンドと SQL ステートメントを Amazon Redshift RSQL と互換性のある形式に変換する方法に関する情報が記載されています。評価レポートには、FastExport コマンドのアクション項目と、AWS SCT が変換できない SQL ステートメントが含まれています。

Teradata FastExport ジョブのスクリプト変換評価レポートを作成するには

1. 左側のパネルの [スクリプト] の下にある [FastExport] ノードを展開します。
2. 変換するスクリプトを選択し、コンテキスト (右クリック) メニューを開いてから、[レポートの作成] を選択します。
3. [概要] タブを表示します。[概要] タブには、FastExport ジョブスクリプト評価レポートからのエグゼクティブサマリー情報が表示されます。これには、ソーススクリプトからのすべての FastExport コマンドと SQL ステートメントの変換結果が含まれます。
4. FastExport ジョブ スクリプト変換評価レポートのローカル コピーを PDF ファイルまたはカンマ区切り値 (CSV) ファイルとして保存できます。

    a. FastExport ジョブスクリプト変換評価レポートを PDF ファイルとして保存するには、右上の [PDF に保存] を選択します。

        PDF ファイルには、スクリプト変換に関するエグゼクティブサマリー、アクション項目、推奨事項が含まれています。

    b. FastExport ジョブスクリプト変換評価レポートを CSV ファイルとして保存するには、右上の [CSV に保存] を選択します。

        CSV ファイルには、アクション項目、推奨アクション、およびスクリプトの変換に必要な推定手作業の複雑さが含まれています。

5. [アクション項目] タブを選択します。このタブには、Amazon Redshift RSQL への手動変換が必要な項目のリストが含まれます。リストからアクション項目を選択すると、アクション項目が適用されるソース FastExport ジョブスクリプトが AWS SCT によって強調表示されます。

AWS SCT で変換された Teradata FastExport ジョブスクリプトの編集と保存

変換したスクリプトは AWS SCT プロジェクトの下部パネルで編集できます。AWS SCT は、プロジェクトの一部として編集されたスクリプトを保存します。

変換されたスクリプトを保存するには

1. ターゲットデータベースパネルの [スクリプト] にある [RSQL スクリプト] ノードを展開します。
2. 変換された、コンテキスト (右クリック) メニューを開いて、[スクリプトの保存] を選択します。
3. 変換したスクリプトを保存するフォルダへのパスを入力し、[保存] を選択します。

   AWS SCT は変換したスクリプトをファイルに保存し、このファイルを開きます。

AWS SCT を使用した Teradata FastLoad ジョブスクリプトの Amazon Redshift RSQL への変換

AWS Schema Conversion Tool(AWS SCT) を使用して Teradata FastLoad ジョブスクリプトを Amazon Redshift RSQL に変換できます。

Teradata FastLoad スクリプトは、複数のセッションを使用して Teradata データベースの空のテーブルにデータをロードする一連のコマンドです。Teradata FastLoad は一連の Teradata FastLoad コマンドと SQL ステートメントを処理します。Teradata FastLoad コマンドは、データ転送のセッション制御とデータ処理を行います。SQL ステートメントはテーブルを作成、管理、削除します。

AWS SCT は、Teradata FastLoad コマンドと SQL ステートメントを Amazon Redshift RSQL と互換性のある形式に変換します。Teradata データベースを Amazon Redshift に移行したら、変換されたこれらのスクリプトを使用して Amazon Redshift データベースにデータをロードできます。

トピック

- FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加する

- Teradata FastLoad ジョブスクリプトの代替変数は、を使用して構成します。AWS SCT
- AWS SCT を使用した Teradata FastLoad ジョブスクリプトの変換
- AWS SCT による Teradata FastLoad ジョブスクリプトの管理
- AWS SCT を使用して Teradata FastLoad ジョブスクリプト変換の評価レポートを作成します。
- AWS SCT で変換された Teradata FastLoad ジョブスクリプトの編集と保存

FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加する

複数のスクリプトを単一の AWS SCT プロジェクトに追加できます。

FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加するには

1. AWS SCT で新しいプロジェクトを作成するか、既存のプロジェクトを開きます。詳細については、「the section called “プロジェクトの作成”」を参照してください。
2. メニューから [ソースを追加] を選択し、次に [Teradata] を選択してソースデータベースをプロジェクトに追加します。詳細については、「ソースとしての Teradata の使用」を参照してください。
3. メニューから [ターゲットを追加] を選択し、ターゲットの Amazon Redshift データベースを AWS SCT プロジェクトに追加します。

   仮想 Amazon Redshift ターゲットデータベースプラットフォームを使用できます。詳細については、「仮想ターゲットの使用」を参照してください。
4. ソース Teradata データベースと Amazon Redshift ターゲットを含む新しいマッピングルールを作成します。詳細については、「新しいマッピングルールの追加」を参照してください。
5. [ビュー] メニューで、[メインビュー] を選択します。
6. 左側のパネルで、[スクリプト] ノードを展開します。
7. [FastLoad] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトのロード] を選択します。
8. ソース Teradata FastLoad ジョブスクリプトの場所を入力し、[フォルダを選択] を選択します。

   AWS SCT[スクリプトのロード] ウィンドウが表示されます。
9. 次のいずれかを実行します。

   - Teradata FastLoad ジョブスクリプトに代替変数が含まれていない場合は、[代替変数なし] を選択し、[OK] を選択してスクリプトを AWS SCT プロジェクトに追加します。

- Teradata FastLoad ジョブスクリプトに代替変数が含まれている場合は、代替変数を設定します。詳細については、「FastLoad ジョブスクリプトでの代替変数の設定」を参照してください。

Teradata FastLoad ジョブスクリプトの代替変数は、を使用して構成します。AWS SCT

Teradata FastLoad ジョブスクリプトには代替変数が含まれている場合があります。たとえば、代替変数を含む 1 つのスクリプトを使用して、データをさまざまなデータベースにロードできます。

代替変数を使用して FastLoad ジョブスクリプトを実行する前に、必ずすべての変数に値を割り当ててください。そのためには、Bash スクリプト、UC4 (Automic) など、他のツールやアプリケーションを使用できます。

AWS SCT は、値を割り当てた後にのみ代替変数を解決および変換できます。ソース Teradata FastLoad ジョブスクリプトの変換を開始する前に、必ずすべての代替変数に値を割り当ててください。AWS SCT を使用して Teradata スクリプトで代替変数を設定できます。

FastLoad ジョブスクリプトで代替変数を設定するには

1. ソース Teradata FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加するときに、[代替変数が使用される] を選択します。これらのスクリプトの追加についての詳細は、「FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加する」をご参照ください。
2. [変数フォーマットの定義] には、スクリプト内のすべての代替変数と一致する正規表現を入力します。

   たとえば、代替変数の名前が ${ で始まり、} で終わる場合は、`\$\{\w+\}` 正規表現を使用します。ドル記号またはパーセント記号で始まる代替変数を照合するには、`\$\w+|\%\w+` 正規表現を使用します。

   AWS SCT の正規表現は、Java の正規表現の構文に準拠しています。詳細については、Java ドキュメントの「java.util.regex クラスパターン」を参照してください。
3. [OK] を選択してスクリプトを AWS SCT プロジェクトにロードし、[OK] を選択して [スクリプトのロード] ウィンドウを閉じます。
4. 左側のパネルで、[スクリプト] ノードを展開します。[FastLoad] を選択し、スクリプトが保存されているフォルダを選択します。コンテキスト (右クリック) メニューを開き、[代替変数] で、[変数をエクスポート] を選択します。

また、1 つのスクリプトの代替変数をエクスポートすることもできます。スクリプトを含むフォルダを展開し、スクリプトを選択し、コンテキスト (右クリック) メニューを開き、[代替変数] で [変数をエクスポート] を選択します。

5. 代替変数を保存するカンマ区切り値 (CSV) ファイルの名前を入力し、[保存] を選択します。
6. この CSV ファイルを開き、代替変数の値を入力します。

   オペレーティング システムに応じて、AWS SCT は CSV ファイルに異なる形式を使用されます。ファイル内の値は、引用符で囲まれていてもいなくてもかまいません。代替変数の値には、ファイル内の他の値と同じ形式を使用してください。AWS SCT は、値が異なる形式の CSV ファイルはインポートできません。
7. CSV ファイルを保存します。
8. 左側のパネルで、[スクリプト] ノードを展開します。[FastLoad] を選択し、スクリプトを選択します。コンテキスト (右クリック) メニューを開き、[代替変数] で、[変数をインポート] を選択します。
9. CSV ファイルを選択し、[開く] を選択します。
10. [変数] を選択すると、検出されたすべての代替変数とその値が表示されます。

AWS SCT を使用した Teradata FastLoad ジョブスクリプトの変換

以下では、AWS SCT を使用して Teradata FastLoad ジョブを Amazon Redshift RSQL に変換する方法について説明します。

Teradata FastLoad ジョブスクリプトを Amazon Redshift RSQL に変換するには

1. FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加します。詳細については、「FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加する」を参照してください。
2. 代替変数を設定します。詳細については、「FastLoad ジョブスクリプトでの代替変数の設定」を参照してください。
3. 左側のパネルで、[スクリプト] ノードを展開します。
4. 次のいずれかを実行します。

   - 1 つの FastLoad ジョブスクリプトを変換するには、[FastLoad] ノードを展開し、変換するスクリプトを選択して、コンテキスト (右クリック) メニューから [スクリプトの変換] を選択します。

- 複数のスクリプトを変換するには、変換するスクリプトをすべて選択してください。[FastLoad] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトの変換] を選択します。次に、以下のいずれかを行ってください。
  - Amazon S3 にソースデータファイルを保存する場合は、[ソースデータファイルの場所] に [S3 オブジェクトパス] を選択します。

    ソースデータファイルのマニフェストファイルの [Amazon S3 バケットフォルダ] と [Amazon S3 バケット] の値を入力します。
  - Amazon S3 にソースデータファイルを保存しない場合は、[ソースデータファイルの場所] に [ホストアドレス] を選択します。

    ホストの [URL または IP アドレス]、[ホストユーザーのログイン名]、およびソースデータファイルの [マニフェストファイルの Amazon S3 バケット] の値を入力します。

5. [OK] をクリックします。

   AWS SCT は、選択したすべての Teradata FastLoad ジョブスクリプトを Amazon Redshift RSQL と互換性のある形式に変換します。変換されたスクリプトは、ターゲットデータベースパネルの [スクリプト] ノードにあります。
6. 変換した Amazon Redshift RSQL スクリプトを編集するか、保存してください。詳細については、「変換された FastLoad ジョブスクリプトの編集と保存」を参照してください。

AWS SCT による Teradata FastLoad ジョブスクリプトの管理

複数の Teradata FastLoad ジョブスクリプトを追加したり、FastLoad ジョブスクリプト AWS SCT をプロジェクトから削除したりできます。

新しい FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加するには

1. 左パネルの [スクリプト] ノードを展開します。
2. [FastLoad] ノードを選択し、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトのロード] を選択します。
4. 新しい FastLoad ジョブスクリプトを追加し、代替変数を設定するのに必要な情報を入力します。詳細については、「FastLoad ジョブスクリプトを AWS SCT プロジェクトに追加する」および「FastLoad ジョブスクリプトでの代替変数の設定」を参照してください。

FastLoad ジョブスクリプトを AWS SCT プロジェクトから削除するには

1. 左側のパネルの [スクリプト] の下にある [FastLoad] ノードを展開します。
2. 削除するスクリプトを選択して、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトを削除] を選択します。

AWS SCT を使用して Teradata FastLoad ジョブスクリプト変換の評価レポートを作成します。

FastLoad ジョブスクリプト変換評価レポートには、FastLoad コマンドと SQL ステートメントの変換に関する情報が記載されています。ソーススクリプトから Amazon Redshift RSQL と互換性のある形式への変換が行われます。評価レポートには、FastLoad コマンドのアクション項目と、AWS SCT が変換できない SQL ステートメントが含まれています。

Teradata FastLoad ジョブのスクリプト変換評価レポートを作成するには

1. 左側のパネルの [スクリプト] の下にある [FastLoad] ノードを展開します。
2. 変換するスクリプトを選択し、コンテキスト (右クリック) メニューを開いてから、[レポートの作成] を選択します。
3. [概要] タブを表示します。

   [概要] タブには、FastLoad ジョブスクリプト評価レポートからのエグゼクティブサマリー情報が表示されます。これには、ソーススクリプトからのすべての FastLoad コマンドと SQL ステートメントの変換結果が含まれます。
4. (オプション) FastLoad ジョブスクリプト変換評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらかとして保存します。

   - FastLoad ジョブスクリプト変換評価レポートを PDF ファイルとして保存するには、右上の [PDF に保存] を選択します。

     PDF ファイルには、スクリプト変換に関するエグゼクティブサマリー、アクション項目、推奨事項が含まれています。
   - FastLoad ジョブスクリプト変換評価レポートを CSV ファイルとして保存するには、右上の [CSV に保存] を選択します。

     CSV ファイルには、アクション項目、推奨アクション、およびスクリプトの変換に必要な推定手作業の複雑さが含まれています。

5. [アクション項目] タブを選択します。このタブには、Amazon Redshift RSQL への手動変換が必要な項目のリストが含まれます。リストからアクション項目を選択すると、アクション項目が適用されるソース FastLoad ジョブスクリプトが AWS SCT によって強調表示されます。

AWS SCT で変換された Teradata FastLoad ジョブスクリプトの編集と保存

変換したスクリプトは AWS SCT プロジェクトの下部パネルで編集できます。AWS SCT は、プロジェクトの一部として編集されたスクリプトを保存します。

変換されたスクリプトを保存するには

1. ターゲットデータベースパネルの [スクリプト] にある [RSQL スクリプト] ノードを展開します。
2. 変換された、コンテキスト (右クリック) メニューを開いて、[スクリプトの保存] を選択します。
3. 変換したスクリプトを保存するフォルダへのパスを入力し、[保存] を選択します。

   AWS SCT は変換したスクリプトをファイルに保存し、このファイルを開きます。

AWS SCT でTeradata MultiLoad ジョブスクリプトから Amazon Redshift RSQL への変換を行う

AWS SCT を使用して、Teradata MultiLoad ジョブスクリプトから Amazon Redshift RSQL に変換できます。

Teradata MultiLoad ジョブスクリプトは、Teradata データベースのバッチメンテナンスを行うためのコマンドセットです。Teradata MultiLoad インポートタスクは、最大 5 つの異なるテーブルとビューに対してさまざまな挿入、更新、削除操作を実行します。Teradata MultiLoad 削除タスクでは、1 つのテーブルから多数の行を削除できます。

AWS SCT は、Teradata MultiLoad コマンドと SQL ステートメントを Amazon Redshift RSQL と互換性のある形式に変換します。Teradata データベースを Amazon Redshift に移行したら、変換されたこれらのスクリプトを使用して Amazon Redshift データベースのデータを管理します。

トピック

- MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加する

- Teradata MultiLoad ジョブスクリプトの代替変数は、AWS SCT を使用して設定します。
- AWS SCT を使用した Teradata マルチロードジョブスクリプトの変換
- AWS SCT での Teradata MultiLoad ジョブスクリプトの管理
- AWS SCT を使用して Teradata MultiLoad ジョブスクリプト変換用の評価レポートを作成します。
- AWS SCT で変換された Teradata MultiLoad ジョブスクリプトの編集と保存

MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加する

複数のスクリプトを単一の AWS SCT プロジェクトに追加できます。

MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加するには

1. AWS SCT で新しいプロジェクトを作成するか、既存のプロジェクトを開きます。詳細については、「the section called “プロジェクトの作成”」を参照してください。
2. メニューから [ソースを追加] を選択し、次に [Teradata] を選択してソースデータベースをプロジェクトに追加します。詳細については、「ソースとしての Teradata の使用」を参照してください。
3. メニューから [ターゲットを追加] を選択し、ターゲットの Amazon Redshift AWS SCT データベースをプロジェクトに追加します。

   仮想 Amazon Redshift ターゲットデータベースプラットフォームを使用できます。詳細については、「仮想ターゲットの使用」を参照してください。
4. ソース Teradata データベースと Amazon Redshift ターゲットを含む新しいマッピングルールを作成します。詳細については、「新しいマッピングルールの追加」を参照してください。
5. [ビュー] メニューで、[メインビュー] を選択します。
6. 左側のパネルで、[スクリプト] ノードを展開します。
7. [MultiLoad] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトのロード] を選択します。
8. ソース Teradata MultiLoad ジョブスクリプトの場所を入力し、[フォルダの選択] を選択します。

   AWS SCT[スクリプトのロード] ウィンドウが表示されます。
9. 次のいずれかを実行します。

   - Teradata MultiLoad ジョブスクリプトに代替変数が含まれていない場合は、[代替変数なし] を選択し、[OK] 選択してスクリプトを AWS SCT プロジェクトに追加します。

- Teradata MultiLoad ジョブスクリプトに代替変数が含まれている場合は、代替変数を設定します。詳細については、「MultiLoad ジョブスクリプトでの代替変数の設定」を参照してください。

Teradata MultiLoad ジョブスクリプトの代替変数は、AWS SCT を使用して設定します。

Teradata マルチロードジョブスクリプトには代替変数が含まれている場合があります。たとえば、代替変数を含む 1 つのスクリプトを使用して、データをさまざまなデータベースにロードできます。

代替変数を使用して MultiLoad ジョブスクリプトを実行する前に、必ずすべての変数に値を割り当ててください。そのためには、Bash スクリプト、UC4 (Automic) など、他のツールやアプリケーションを使用できます。

AWS SCT は、値を割り当てた後にのみ代替変数を解決および変換できます。ソース Teradata MultiLoad ジョブスクリプトの変換を開始する前に、すべての代替変数に値を割り当てていることを確認してください。AWS SCT を使用して Teradata スクリプトで代替変数を設定できます。

MultiLoad ジョブスクリプトで代替変数を設定するには

1. ソース Teradata MultiLoad AWS SCT ジョブスクリプトをプロジェクトに追加するときに、[代替変数が使用される] を選択します。これらのスクリプトの追加についての詳細は、「MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加する」をご参照ください。
2. [変数フォーマットの定義] には、スクリプト内のすべての代替変数と一致する正規表現を入力します。

   たとえば、代替変数の名前が ${ で始まり、} で終わる場合は、`\$\{\w+\}` 正規表現を使用します。ドル記号またはパーセント記号で始まる代替変数を照合するには、`\$\w+|\%\w+` 正規表現を使用します。

   AWS SCT の正規表現は、Java の正規表現の構文に準拠しています。詳細については、Java ドキュメントの「java.util.regex クラスパターン」を参照してください。
3. [OK] を選択してスクリプトを AWS SCT プロジェクトにロードし、[OK] を選択して [スクリプトのロード] ウィンドウを閉じます。
4. [変数] を選択すると、検出されたすべての代替変数とその値が表示されます。
5. [値] には、代替変数の値を入力します。

AWS SCT を使用した Teradata マルチロードジョブスクリプトの変換

以下では、AWS SCT を使用して Teradata マルチロードジョブを Amazon Redshift RSQL に変換する方法について説明します。

Teradata MultiLoad ジョブスクリプトを Amazon Redshift RSQL に変換するには

1. MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加します。詳細については、「MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加する」を参照してください。
2. 代替変数を設定し、その値を入力します。詳細については、「MultiLoad ジョブスクリプトでの代替変数の設定」を参照してください。
3. 左側のパネルで、[スクリプト] ノードを展開します。
4. 次のいずれかを実行します。

   - 1 つの MultiLoad ジョブスクリプトを変換するには、[MultiLoad] ノードを展開し、変換するスクリプトを選択して、コンテキスト (右クリック) メニューから [スクリプトの変換] を選択します。
   - 複数のスクリプトを変換するには、変換するスクリプトをすべて選択してください。[MultiLoad] を選択し、コンテキスト (右クリック) メニューを開いてから、[スクリプトの変換] を選択します。
5. 次のいずれかを実行します。

   - Amazon S3 にソースデータファイルを保存する場合は、[ソースデータファイルの場所] に [S3 オブジェクトパス] を選択します。

     ソースデータファイルのマニフェストファイル用の Amazon S3 バケットフォルダと Amazon S3 バケットを入力します。
   - Amazon S3 にソースデータファイルを保存しない場合は、[ソースデータファイルの場所] に [ホストアドレス] を選択します。

     ホストの URL または IP アドレス、ホストユーザーのログイン名、およびソースデータファイルのマニフェストファイルの Amazon S3 バケットを入力します。
6. [OK] をクリックします。

   AWS SCT は、選択したすべての Teradata MultiLoad ジョブスクリプトを Amazon Redshift RSQL と互換性のある形式に変換します。変換されたスクリプトは、ターゲットデータベースパネルの [スクリプト] ノードにあります。

7. 変換した Amazon Redshift RSQL スクリプトを編集するか、保存してください。詳細については、「変換された MultiLoad ジョブスクリプトの編集と保存」を参照してください。

AWS SCT での Teradata MultiLoad ジョブスクリプトの管理

複数の Teradata MultiLoad ジョブスクリプトを追加したり、MultiLoad ジョブスクリプトを AWS SCT プロジェクトから削除したりできます。

新しい MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加するには

1. 左パネルの [スクリプト] ノードを展開します。
2. [MultiLoad] ノードを選択し、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトのロード] を選択します。
4. 新しい MultiLoad ジョブスクリプトを追加し、代替変数を設定するのに必要な情報を入力します。詳細については、「MultiLoad ジョブスクリプトを AWS SCT プロジェクトに追加する」および「MultiLoad ジョブスクリプトでの代替変数の設定」を参照してください。

MultiLoad ジョブスクリプトを AWS SCT プロジェクトから削除するには

1. 左側のパネルの [スクリプト] にある [MultiLoad] ノードを展開します。
2. 削除するスクリプトを選択して、コンテキスト (右クリック) メニューを開きます。
3. [スクリプトを削除] を選択します。

AWS SCT を使用して Teradata MultiLoad ジョブスクリプト変換用の評価レポートを作成します。

MultiLoad ジョブスクリプト変換評価レポートには、MultiLoad コマンドと SQL ステートメントの変換に関する情報が記載されています。この変換は、ソーススクリプトから Amazon Redshift の RSQL コマンドと Amazon Redshift の SQL ステートメントに変換されます。評価レポートには、MultiLoad コマンドのアクション項目と、AWS SCT で変換できない SQL ステートメントが含まれます。

Teradata MultiLoad ジョブのスクリプト変換評価レポートを作成するには

1. 左側のパネルの [スクリプト] にある [MultiLoad] ノードを展開します。

2. 評価レポートの作成対象のスクリプトを選択し、コンテキスト (右クリック) メニューを開いてから、[レポートの作成] を選択します。
3. [概要] タブを表示します。[概要] タブには、MultiLoad ジョブスクリプト評価レポートからのエグゼクティブサマリー情報が表示されます。これには、ソーススクリプトからのすべての MultiLoad コマンドと SQL ステートメントの変換結果が含まれます。
4. (オプション) MultiLoad ジョブスクリプト変換評価レポートのローカルコピーを PDF ファイルまたはカンマ区切り値 (CSV) のどちらかとして保存します。

   - MultiLoad ジョブスクリプト変換評価レポートを PDF ファイルとして保存するには、右上の [PDF に保存] を選択します。

     PDF ファイルには、スクリプト変換に関するエグゼクティブサマリー、アクション項目、推奨事項が含まれています。
   - MultiLoad ジョブスクリプト変換評価レポートを CSV ファイルとして保存するには、右上の [CSV に保存] を選択します。

     AWS SCT は 2 つの CSV ファイルを作成します。これらのファイルには、エグゼクティブサマリー、アクション項目、推奨アクション、およびスクリプトの変換に必要な推定手作業の複雑さが含まれます。
5. [アクション項目] タブを選択します。このタブには、Amazon Redshift RSQL への手動変換が必要な項目のリストが含まれます。リストからアクション項目を選択すると、アクション項目が適用されるソース MultiLoad ジョブスクリプトが AWS SCT によって強調表示されます。

AWS SCT で変換された Teradata MultiLoad ジョブスクリプトの編集と保存

変換したスクリプトは AWS SCT プロジェクトの下部パネルで編集できます。AWS SCT は、プロジェクトの一部として編集されたスクリプトを保存します。

変換されたスクリプトを保存するには

1. ターゲットデータベースパネルの [スクリプト] にある [RSQL スクリプト] ノードを展開します。
2. 変換された、コンテキスト (右クリック) メニューを開いて、[スクリプトの保存] を選択します。
3. 変換したスクリプトを保存するフォルダへのパスを入力し、[保存] を選択します。

AWS SCT は変換したスクリプトをファイルに保存し、このファイルを開きます。

AWS Schema Conversion Tool によるビッグデータフレームワークの移行

AWS Schema Conversion Tool(AWS SCT) を使用して、ビッグデータフレームワークを AWS クラウド に移行できます。

現在、AWS SCT は Hadoop クラスターの Amazon EMR および Amazon S3 への移行をサポートしています。この移行プロセスには Hive サービスと HDFS サービスが含まれます。

また、AWS SCT を使用して Apache Oozie オーケストレーションワークフローから AWS Step Functions への変換を自動化することもできます。

トピック

- AWS Schema Conversion Tool を使用して Apache Hadoop を Amazon EMR に移行する
- AWS Schema Conversion Tool を使用して Apache Oozie を AWS Step Functions に変換する

AWS Schema Conversion Tool を使用して Apache Hadoop を Amazon EMR に移行する

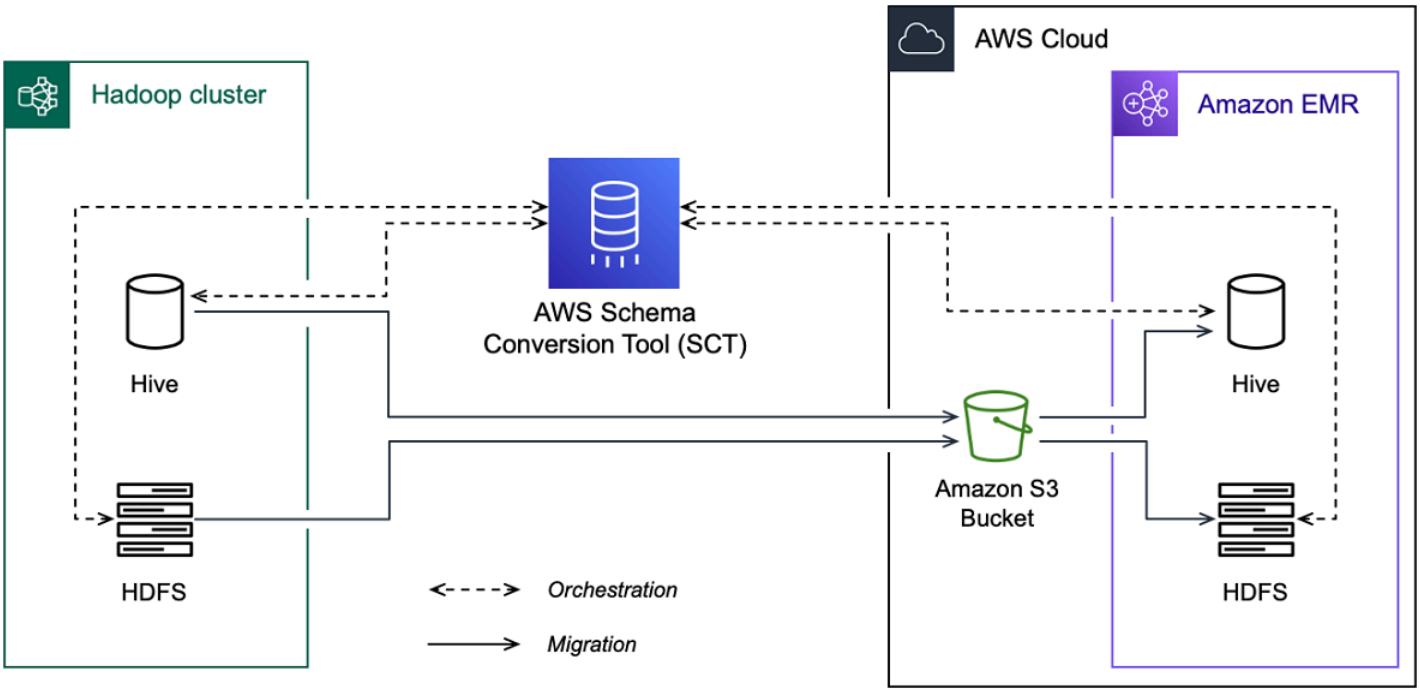
Apache Hadoop クラスターを移行するには、必ず AWS SCT バージョン 1.0.670 以降を使用してください。また、AWS SCT のコマンドラインインターフェイス (CLI) についても理解しておいてください。詳細については、「AWS SCT CLI リファレンス」を参照してください。

トピック

- 移行の概要
- ステップ 1: Hadoop クラスターに接続する
- ステップ 2: マッピングルールを設定する
- ステップ 3: 評価レポートを作成する
- ステップ 4: AWS SCT を使用して Apache Hadoop クラスターを Amazon EMR に移行する
- CLI スクリプトの実行
- ビッグデータ移行プロジェクトの管理

移行の概要

以下の画像は、Apache Hadoop から Amazon EMR への移行のアーキテクチャ図を示しています。



AWS SCT は、ソース Hadoop クラスターから Amazon S3 バケットにデータとメタデータを移行します。次に、AWS SCT がソースの Hive メタデータを使用して、ターゲット Amazon EMR Hive サービスにデータベースオブジェクトを作成します。オプションで、AWS Glue Data Catalog をメタストアとして使用するように Hive を設定できます。この場合、AWS SCT は Hive のソースメタデータを AWS Glue Data Catalog に移行します。

その後、AWS SCT を使用して Amazon S3 バケットからターゲットの Amazon EMR HDFS サービスにデータを移行できます。または、データを Amazon S3 バケットに残して、Hadoop ワークロードのデータリポジトリとして使用することもできます。

Hapood の移行を開始するには、AWS SCT CLI スクリプトを作成して実行します。このスクリプトには、移行を実行するためのコマンド一式が含まれています。Hadoop 移行スクリプトのテンプレートをダウンロードして編集できます。詳細については、「CLI シナリオの取得」を参照してください。

Apache Hadoop から Amazon S3 と Amazon EMR への移行を実行できるように、スクリプトに次のステップが含まれていることを確認してください。

ステップ 1: Hadoop クラスターに接続する

Apache Hadoop クラスターの移行を開始するには、新しい AWS SCT プロジェクトを作成します。次に、ソースクラスターとターゲットクラスターに接続します。移行を開始する前に、必ずターゲット AWS リソースを作成してプロビジョニングしてください。

このステップでは、次の AWS SCT CLI コマンドを使用します。

- `CreateProject` – 新しい AWS SCT プロジェクトを作成します。
- `AddSourceCluster` – AWS SCT プロジェクト内のソース Hadoop クラスターに接続します。
- `AddSourceClusterHive` – プロジェクト内のソース Hive サービスに接続します。
- `AddSourceClusterHDFS` – プロジェクト内のソース HDFS サービスに接続します。
- `AddTargetCluster` – プロジェクト内のターゲット Amazon EMR クラスターに接続します。
- `AddTargetClusterS3` – Amazon S3 バケットをプロジェクトに追加します。
- `AddTargetClusterHive` – プロジェクト内のターゲット Hive サービスに接続します。
- `AddTargetClusterHDFS` – プロジェクト内のターゲット HDFS サービスに接続します。

これらの AWS SCT コマンドの使用例については、「Apache Hadoop をソースとして使用」を参照してください。

ソースまたはターゲットクラスターに接続するコマンドを実行すると、AWS SCT はこのクラスターへの接続を確立しようとします。接続に失敗すると、AWS SCT で CLI スクリプトからのコマンドの実行が中止され、エラーメッセージが表示されます。

ステップ 2: マッピングルールを設定する

ソースクラスターとターゲットクラスターに接続したら、マッピングルールを設定します。マッピングルールは、ソースクラスターの移行ターゲットを定義します。AWS SCT プロジェクトに追加したすべてのソースクラスターのマッピングルールを必ず設定してください。テーブルマッピングの詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。

このステップでは、`AddServerMapping` コマンドを使用します。このコマンドは、ソースクラスターとターゲットクラスターを定義する 2 つのパラメータを使用します。この `AddServerMapping` コマンドは、データベースオブジェクトへの明示的なパスまたはオブジェクト名とともに使用できます。1 つ目のオプションには、オブジェクトのタイプと名前を指定します。2 つ目のオプションには、オブジェクト名のみを指定します。

- sourceTreePath — ソースデータベースオブジェクトへの明示的なパス。

  targetTreePath — ターゲットデータベースオブジェクトへの明示的なパス。
- sourceNamePath — ソースオブジェクトの名前のみを含むパス。

  targetNamePath — ターゲットオブジェクトの名前のみを含むパス。

次のコード例では、ソース testdb Hive データベースとターゲット EMR クラスターの明示的なパスを使用してマッピングルールを作成します。

```
AddServerMapping
 -sourceTreePath: 'Clusters.HADOOP_SOURCE.HIVE_SOURCE.Databases.testdb'
 -targetTreePath: 'Clusters.HADOOP_TARGET.HIVE_TARGET'
/
```

Windows では、この例と以下の例を使用できます。Linux で CLI コマンドを実行するには、使用しているオペレーティングシステムに合わせてファイルパスを適切に更新してください。

次のコード例では、オブジェクト名のみを含むパスを使用してマッピングルールを作成します。

```
AddServerMapping
 -sourceNamePath: 'HADOOP_SOURCE.HIVE_SOURCE.testdb'
 -targetNamePath: 'HADOOP_TARGET.HIVE_TARGET'
/
```

ソースオブジェクトのターゲットとして Amazon EMR または Amazon S3 を選択できます。ソースオブジェクトごとに、1 つの AWS SCT プロジェクト内のターゲットを 1 つのみ選択できます。ソースオブジェクトの移行ターゲットを変更するには、既存のマッピングルールを削除し、新しいマッピングルールを作成します。マッピングルールを削除するには、DeleteServerMapping コマンドを使用します。このコマンドは、次の 2 つのパラメータのうちの 1 つを使用します。

- sourceTreePath — ソースデータベースオブジェクトへの明示的なパス。
- sourceNamePath — ソースオブジェクトの名前のみを含むパス。

AddServerMapping および DeleteServerMapping コマンドの詳細については、『AWS Schema Conversion Tool CLI リファレンス』を参照してください。

ステップ 3: 評価レポートを作成する

移行を開始する前に、評価レポートを作成することをお勧めします。このレポートには、すべての移行タスクがまとめられ、移行中に明らかになるアクションアイテムが詳しく説明されています。移行が失敗しないようにするには、このレポートを参照して、移行前のアクションアイテムに対処してください。詳細については、「移行評価レポート」を参照してください。

このステップでは、CreateMigrationReport コマンドを使用します。このコマンドは、2 つのパラメータを使用します。treePath パラメーターは必須で、forceMigrate パラメーターはオプションです。

- treePath — 評価レポートのコピーを保存するソースデータベースオブジェクトへの明示的なパス。
- forceMigrate — true に設定すると、同じオブジェクトを参照する HDFS フォルダと Hive テーブルがプロジェクトに含まれていても AWS SCT は移行を続行します。デフォルト値は、「false」です。

その後、評価レポートのコピーを PDF またはカンマ区切り (CSV) ファイルとして保存できます。これを行うには、SaveReportPDF コマンドまたは SaveReportCSV コマンドを使用します。

SaveReportPDF コマンドは、評価レポートのコピーを PDF ファイルとして保存します。このコマンドは、4 つのパラメータを使用します。file パラメータは必須です。他のパラメータはオプションです。

- file — PDF ファイルへのパスとその名前。
- filter — 移行するソースオブジェクトの範囲を定義するために以前に作成したフィルタの名前。
- treePath — 評価レポートのコピーを保存するソースデータベースオブジェクトへの明示的なパス。
- namePath — 評価レポートのコピーを保存する対象オブジェクトの名前のみを含むパス。

SaveReportCSV コマンドは、評価レポートを 3 つの CSV ファイルに保存します。このコマンドは、4 つのパラメータを使用します。directory パラメータは必須です。他のパラメータはオプションです。

- directory — AWS SCT が CSV ファイルを保存するフォルダへのパス。

- filter — 移行するソースオブジェクトの範囲を定義するために以前に作成したフィルタの名前。
- treePath — 評価レポートのコピーを保存するソースデータベースオブジェクトへの明示的なパス。
- namePath — 評価レポートのコピーを保存する対象オブジェクトの名前のみを含むパス。

次のコード例では、評価レポートのコピーを c:\sct\ar.pdf ファイルに保存しています。

```
SaveReportPDF
 -file:'c:\sct\ar.pdf'
/
```

次のコード例では、評価レポートのコピーを CSV ファイルとして c:\sct フォルダに保存します。

```
SaveReportCSV
 -file:'c:\sct'
/
```

SaveReportPDF および SaveReportCSV コマンドの詳細については、『AWS Schema Conversion Tool CLI リファレンス』を参照してください。

ステップ 4: AWS SCT を使用して Apache Hadoop クラスターを Amazon EMR に移行する

AWS SCT プロジェクトを設定したら、オンプレミスの Apache Hadoop クラスターの AWS クラウド への移行を開始します。

このステップでは、Migrate、MigrationStatus、ResumeMigration コマンドを使用します。

Migrate コマンドは、ソースオブジェクトをターゲットクラスターに移行します。このコマンドは、4 つのパラメータを使用します。必ず filter または treePathパラメータを指定してください。その他のパラメータは省略可能です。

- filter — 移行するソースオブジェクトの範囲を定義するために以前に作成したフィルタの名前。
- treePath — 評価レポートのコピーを保存するソースデータベースオブジェクトへの明示的なパス。

- forceLoad — true に設定すると、AWS SCT は移行中にデータベースのメタデータツリーを自動的にロードします。デフォルト値は、「false」です。
- forceMigrate — true に設定すると、同じオブジェクトを参照する HDFS フォルダと Hive テーブルがプロジェクトに含まれていても AWS SCT は移行を続行します。デフォルト値は、「false」です。

MigrationStatus コマンドは、移行の進捗に関する情報を返します。このコマンドを実行するには、name パラメータに移行プロジェクトの名前を入力します。この名前は CreateProject コマンドで指定しました。

ResumeMigration コマンドは、Migrate コマンドを使用して起動した中断された移行を再開します。ResumeMigration コマンドはパラメータを使用しません。移行を再開するには、ソースクラスターとターゲットクラスターに接続する必要があります。詳細については、「移行プロジェクトの管理」を参照してください。

次のコード例では、ソース HDFS サービスから Amazon EMR にデータを移行します。

```
Migrate
 -treePath: 'Clusters.HADOOP_SOURCE.HDFS_SOURCE'
 -forceMigrate: 'true'
/
```

CLI スクリプトの実行

AWS SCT CLI スクリプトの編集が終了したら、.scts 拡張子の付いたファイルとして保存します。これで、AWS SCT インストールパスの app フォルダからスクリプトを実行できます。そのためには、次のコマンドを使用します。

```
RunSCTBatch.cmd --pathtoscts "C:\script_path\hadoop.scts"
```

前の例では、*script_path* を CLI スクリプトを含むファイルへのパスに置き換えます。AWS SCT でのスクリプトの実行の詳細については、「スクリプトモード」を参照してください。

ビッグデータ移行プロジェクトの管理

移行が完了したら、AWS SCT プロジェクトを保存して、後で編集することができます。

AWS SCT プロジェクトを保存するには、SaveProject コマンドを使用します。このコマンドはパラメータを使用しません。

次のコード例では AWS SCT プロジェクトを保存します。

```
SaveProject
/
```

AWS SCT プロジェクトを開くには、OpenProject コマンドを使用します。このコマンドは必須パラメータを 1 つ使用します。file パラメーターには、AWS SCT プロジェクトファイルへのパスと名前を入力します。CreateProject コマンドでプロジェクト名を指定しました。OpenProject コマンドを実行するには、.scts プロジェクトファイルの名前に拡張子を必ず追加してください。

次のコード例では、c:\sct フォルダから hadoop_emr プロジェクトを開きます。

```
OpenProject
 -file: 'c:\sct\hadoop_emr.scts'
/
```

AWS SCT プロジェクトを開いたら、ソースクラスターとターゲットクラスターはプロジェクトに追加済みなので、追加する必要はありません。ソースクラスターとターゲットクラスターでの作業を開始するには、それらに接続する必要があります。これを行うには、ConnectSourceCluster および ConnectTargetCluster コマンドを使用します。これらのコマンドは、AddSourceCluster および AddTargetCluster コマンドと同じパラメータを使用します。CLI スクリプトを編集し、これらのコマンドの名前を置き換えてもパラメータのリストは変更されません。

次のコード例では、ソース Hadoop クラスターに接続します。

```
ConnectSourceCluster
    -name: 'HADOOP_SOURCE'
    -vendor: 'HADOOP'
    -host: 'hadoop_address'
    -port: '22'
    -user: 'hadoop_user'
    -password: 'hadoop_password'
    -useSSL: 'true'
    -privateKeyPath: 'c:\path\name.pem'
    -passPhrase: 'hadoop_passphrase'
/
```

次のコード例では、ターゲットの Amazon EMR クラスターに接続します。

```
ConnectTargetCluster
```

```
 -name: 'HADOOP_TARGET'
 -vendor: 'AMAZON_EMR'
 -host: 'ec2-44-44-55-66.eu-west-1.EXAMPLE.amazonaws.com'
 -port: '22'
 -user: 'emr_user'
 -password: 'emr_password'
 -useSSL: 'true'
 -privateKeyPath: 'c:\path\name.pem'
 -passPhrase: '1234567890abcdef0!'
 -s3Name: 'S3_TARGET'
 -accessKey: 'AKIAIOSFODNN7EXAMPLE'
 -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
 -region: 'eu-west-1'
 -s3Path: 'doc-example-bucket/example-folder'
/
```

前の例では、*hadoop_address* を Hadoop クラスターの IP アドレスに置き換えてください。必要に応じて、ポート変数の値を設定します。次に、*hadoop_user* と hadoop_password *##Hadoop* ユーザーの名前とこのユーザーのパスワードに置き換えます。*path\name* には、ソース Hadoop クラスターの PEM ファイルの名前とパスを入力します。ソースクラスターとターゲットクラスターの追加の詳細情報については、「Apache Hadoop を AWS SCT のソースとして使用」を参照してください。

ソースとターゲットの Hadoop クラスターに接続したら、Amazon S3 バケットだけでなく、Hive と HDFS サービスにも接続する必要があります。これを行うには、ConnectSourceClusterHive、ConnectSourceClusterHdfs、ConnectTargetClusterHive、コマンドを使用します。これらのコマンドは、Hive サービス、HDFS サービス、および Amazon S3 バケットをプロジェクトに追加する際に使用したコマンドと同じパラメータを使用します。CLI スクリプトを編集して、コマンド名の Add プレフィックスを Connect に置き換えます。

AWS Schema Conversion Tool を使用して Apache Oozie を AWS Step Functions に変換する

Apache Oozie ワークフローを変換するには、必ず AWS SCT バージョン 1.0.671 以降を使用してください。また、AWS SCT のコマンドラインインターフェイス (CLI) についても理解しておいてください。詳細については、「AWS SCT CLI リファレンス」を参照してください。

トピック

- 変換の概要

- ステップ 1: ソースサービスとターゲットサービスConnect する
- ステップ 2: マッピングルールを設定する
- ステップ 3: パラメータを設定する
- ステップ 4: 評価レポートを作成する
- ステップ 5: AWS SCT を使用して Apache Oozie ワークフローを AWS Step Functions に変換する
- CLI スクリプトの実行
- AWS SCT が AWS Step Functions に変換できる Apache Oozie ノード

変換の概要

Apache Oozie ソースコードには、アクションノード、制御フローノード、ジョブプロパティが含まれます。アクションノードは Apache Oozie ワークフローで実行するジョブを定義します。Apache Oozie を使用して Apache Hadoop クラスターをオーケストレーションする場合、アクションノードには Hadoop ジョブが含まれます。制御フローノードは、ワークフローパスを制御するメカニズムを提供します。制御フローノードには`start`、`end`、`decision`、`fork`、`join` などのノードが含まれます。

AWS SCT は、ソースアクションノードとコントロールフローノードを AWS Step Functions に変換します。AWS Step Functions では、Amazon ステートメント言語 (ASL) でワークフローを定義します。 AWS SCT は ASL を使用して、作業を実行したり、次に移行する状態を決定したり、エラーで停止したりできる状態のコレクションであるステートマシンを定義します。次に、AWS SCT はステートマシンの定義を含む JSON ファイルをアップロードします。次に、AWS SCT は AWS Identity and Access Management (IAM) ロールを使用してステートマシンを AWS Step Functions で設定できます。詳細については、AWS Step Functions デベロッパーガイドの「AWS Step Functions とは」を参照してください。

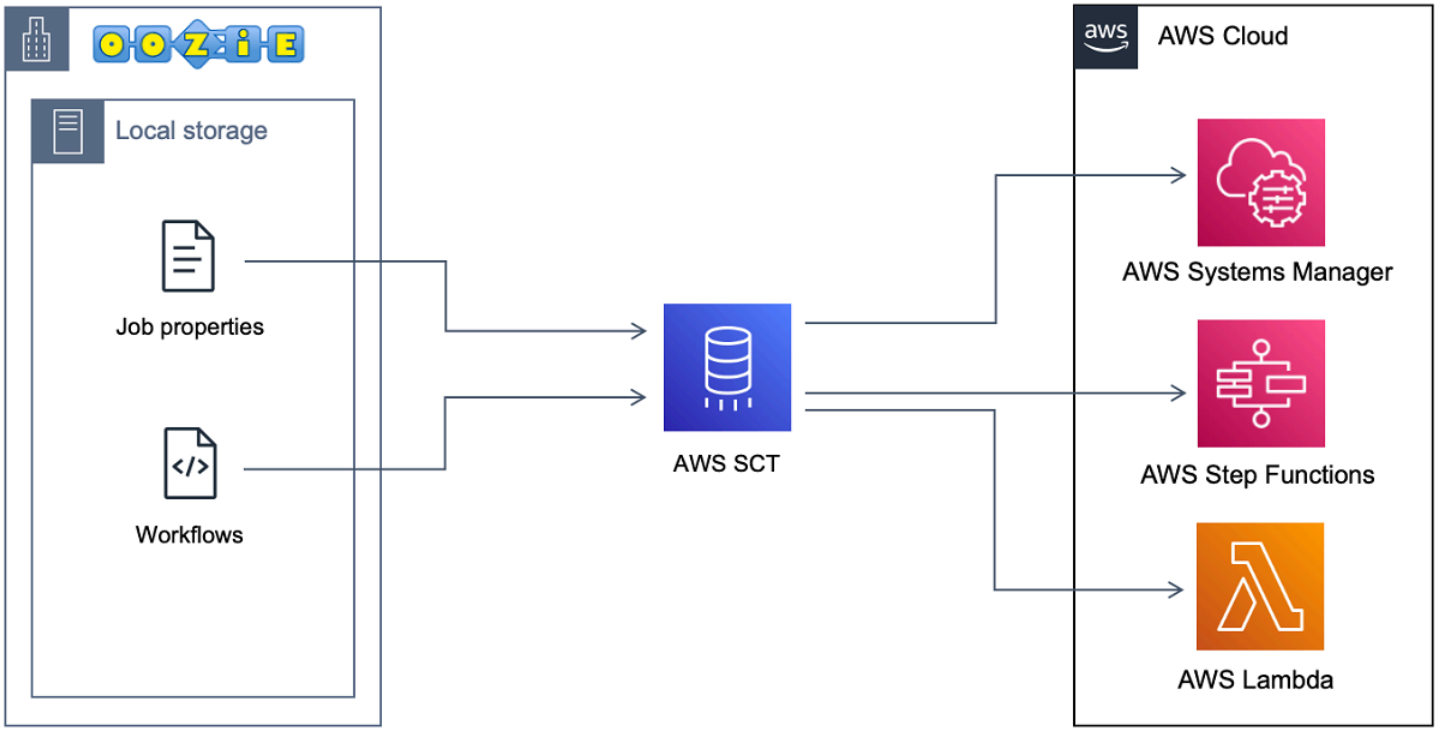
また、AWS SCT は AWS Step Functions がサポートしていないソース関数をエミュレートする AWS Lambda 関数を含む拡張パックを作成します。詳細については、「AWS SCT 拡張パックの使用」を参照してください。

AWS SCT は、ソースジョブのプロパティを AWS Systems Manager に移行します。パラメータの名前と値を保存するには、AWS SCT は AWS Systems Manager の一機能である Parameter Store を使用します。詳細については、『AWS Systems Manager ユーザーガイド』の「AWS Systems Manager とは何か」を参照してください。

AWS SCT を使用して、パラメータの値と名前を自動的に更新できます。Apache Oozie と AWS Step Functions のアーキテクチャの違いにより、パラメータの設定が必要な場合があります。 AWS

SCT は、指定したパラメータ名または値をソースファイル内で検索し、新しい値に置き換えることができます。詳細については、「ステップ 3: パラメータを設定する」を参照してください。

次の図は、Apache Oozie を AWS Step Functions に変換するアーキテクチャ図を示しています。



変換を開始するには、AWS SCT CLI スクリプトを作成して実行します。このスクリプトには、変換を実行するためのコマンド一式が含まれています。Apache Oozie 変換スクリプトのテンプレートをダウンロードして編集できます。詳細については、「CLI シナリオの取得」を参照してください。

スクリプトに次の手順が含まれていることを確認します。

ステップ 1: ソースサービスとターゲットサービスConnect する

Apache Oozie クラスタの変換を開始するには、新しい AWS SCT プロジェクトを作成します。次に、ソースサービスとターゲットサービスに接続します。移行を開始する前に、必ずターゲット AWS リソースを作成してプロビジョニングしてください。詳細については、「Apache Oozie をソースとして使用する場合の前提条件」を参照してください。

このステップでは、次の AWS SCT CLI コマンドを使用します。

- `CreateProject` – 新しい AWS SCT プロジェクトを作成します。
- `AddSource` — ソースの Apache Oozie ファイルを AWS SCT プロジェクトに追加します。
- `ConnectSource` — ソースとして Apache Oozie に接続するためのものです。

- AddTarget — プロジェクトの移行ターゲットとして AWS Step Functions を追加します。
- ConnectTarget – AWS Step Functions に接続します。

これらの AWS SCT コマンドの使用例については、「Apache Oozie をソースとしての使用」を参照してください。

ConnectSource または ConnectTarget コマンドを実行すると、AWS SCT はサービスへの接続を確立しようとします。接続に失敗すると、AWS SCT で CLI スクリプトからのコマンドの実行が中止され、エラーメッセージが表示されます。

ステップ 2: マッピングルールを設定する

ソースサービスとターゲットサービスに接続したら、マッピングルールを設定します。マッピングルールは、ソース Apache Oozie ワークフローとパラメータの移行ターゲットを定義します。テーブルマッピングの詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。

変換するソースオブジェクトとターゲットオブジェクトを定義するには、AddServerMapping コマンドを使用します。このコマンドでは、sourceTreePath と targetTreePath 2 つのパラメータを使用します。これらのパラメータの値には、ソースオブジェクトとターゲットオブジェクトへの明示的なパスが含まれます。Apache Oozie を AWS Step Functions に変換するには、これらのパラメータは ETL で始まる必要があります。

次のコード例では、OOZIE オブジェクトと AWS_STEP_FUNCTIONS オブジェクトのマッピングルールを作成しています。これらのオブジェクトは、前のステップで AddSource および AddTarget コマンドを使用して AWS SCT プロジェクトに追加しました。

```
AddServerMapping
    -sourceTreePath: 'ETL.APACHE_OOZIE'
    -targetTreePath: 'ETL.AWS_STEP_FUNCTIONS'
/
```

AddServerMapping コマンドの使用の詳細については、『AWS Schema Conversion Tool CLI リファレンス』を参照してください。

ステップ 3: パラメータを設定する

ソース Apache Oozie ワークフローでパラメータを使用している場合は、AWS Step Functions への変換後にパラメータの値を変更しなければならない場合があります。また、AWS Step Functions で使用する新しいパラメータを追加する必要がある場合があります。

このステップでは、AddParameterMapping および AddTargetParameter コマンドを使用します。

ソースファイル内のパラメータ値を置き換えるには、AddParameterMapping コマンドを使用します。 AWS SCT はソースファイルをスキャンし、名前または値でパラメータを検索し、値を変更します。1 つのコマンドを実行してすべてのソースファイルをスキャンできます。スキャンするファイルの範囲は、以下のリストの最初の 3 つのパラメータのいずれかを使用して定義します。このコマンドは最大で 6 つのパラメータを使用します。

- filterName — ソースオブジェクトのフィルターの名前。CreateFilter コマンドを使用してフィルタを作成できます。
- treePath — ソースオブジェクトへの明示的なパス。
- namePath — 特定のソースオブジェクトへの明示的なパス。
- sourceParameterName — ソースパラメータの名前。
- sourceValue — ソースパラメータの値。
- targetValue — ターゲットパラメータの値。

次のコード例では、c:\oozie\hive.py 値が等しいすべてのパラメータを s3://bucket-oozie/hive.py 値に置き換えます。

```
AddParameterMapping
 -treePath: 'ETL.OOZIE.Applications'
 -sourceValue: 'c:\oozie\hive.py'
 -targetValue: 's3://bucket-oozie/hive.py'
/
```

次のコード例では、名前が nameNode と同じであるすべてのパラメーターを hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020 値に置き換えます。

```
AddParameterMapping
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -targetValue: 'hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020'
/
```

次のコード例では、名前が nameNode と同じで、値が hdfs://ip-55.eu-west-1.compute.internal:8020 と等しいすべてのパラメータを targetValue パラメータの値に置き換えます。

```
AddParameterMapping
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -sourceValue: 'hdfs://ip-55-66-77-88.eu-west-1.compute.internal:8020'
    -targetValue: 'hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020'
/
```

ソースファイルの既存のパラメータに加えて、ターゲットファイルに新しいパラメータを追加するには、AddTargetParameter コマンドを使用します。このコマンドは、AddParameterMapping コマンドと同じパラメータセットを使用します。

次のコード例では、nameNode パラメーターの代わりに clusterId ターゲットパラメーターを追加しています。

```
AddTargetParameter
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -sourceValue: 'hdfs://ip-55-66-77-88.eu-west-1.compute.internal:8020'
    -targetParameter: 'clusterId'
    -targetValue: '1234567890abcdef0'
/
```

AddServerMapping、AddParameterMapping、AddTargetParameter、CreateFilter コマンドの詳細については、『AWS Schema Conversion Tool CLI リファレンス』を参照してください。

ステップ 4: 評価レポートを作成する

変換を開始する前に、評価レポートを作成することをおすすめします。このレポートには、すべての移行タスクがまとめられ、移行中に明らかになるアクションアイテムが詳しく説明されています。移行が失敗しないようにするには、このレポートを参照して、移行前のアクションアイテムに対処してください。詳細については、「移行評価レポート」を参照してください。

このステップでは、CreateReport コマンドを使用します。このコマンドは、2 つのパラメータを使用します。1 つ目のパラメータは、AWS SCT が評価レポートを作成するソースオブジェクトを指定します。そのためには、filterName、treePath、または namePath のいずれかのパラメーターを使用します。このパラメータは必須です。オプションの Boolean パラメータ forceLoad を追加することもできます。このパラメータを true に設定すると、AWS SCT は、CreateReport コマンドで指定したソースオブジェクトのすべての子オブジェクトを自動的にロードします。

次のコード例では、ソース Oozie ファイルの Applications ノードの評価レポートを作成します。

```
CreateReport
    -treePath: 'ETL.APACHE_OOZIE.Applications'
/
```

その後、評価レポートのコピーを PDF またはカンマ区切り (CSV) ファイルとして保存できます。これを行うには、SaveReportPDF コマンドまたは SaveReportCSV コマンドを使用します。

SaveReportPDF コマンドは、評価レポートのコピーを PDF ファイルとして保存します。このコマンドは、4 つのパラメータを使用します。file パラメータは必須です。他のパラメータはオプションです。

- file — PDF ファイルへのパスとその名前。
- filter — 移行するソースオブジェクトの範囲を定義するために以前に作成したフィルタの名前。
- treePath — 評価レポートのコピーを保存するソースデータベースオブジェクトへの明示的なパス。
- namePath — 評価レポートのコピーを保存する対象オブジェクトの名前のみを含むパス。

SaveReportCSV コマンドは、評価レポートを CSV ファイルに保存します。このコマンドは、4 つのパラメータを使用します。directory パラメータは必須です。他のパラメータはオプションです。

- directory — AWS SCT が CSV ファイルを保存するフォルダへのパス。
- filter — 移行するソースオブジェクトの範囲を定義するために以前に作成したフィルタの名前。
- treePath — 評価レポートのコピーを保存するソースデータベースオブジェクトへの明示的なパス。
- namePath — 評価レポートのコピーを保存する対象オブジェクトの名前のみを含むパス。

次のコード例では、評価レポートのコピーを c:\sct\ar.pdf ファイルに保存しています。

```
SaveReportPDF
 -file:'c:\sct\ar.pdf'
/
```

次のコード例では、評価レポートのコピーを CSV ファイルとして c:\sct フォルダに保存します。

```
SaveReportCSV
 -file:'c:\sct'
/
```

CreateReport、SaveReportPDF、SaveReportCSV コマンドの使用の詳細については、『AWS Schema Conversion Tool CLI コマンドリファレンス』を参照してください。

ステップ 5: AWS SCT を使用して Apache Oozie ワークフローを AWS Step Functions に変換する

AWS SCT プロジェクトを設定したら、ソースコードを変換して AWS クラウド に適用します。

このステップでは、 Convert、SaveOnS3、ConfigureStateMachine、ApplyToTarget コマンドを使用します。

Migrate コマンドは、ソースオブジェクトをターゲットクラスターに移行します。このコマンドは、4 つのパラメータを使用します。必ず filter または treePathパラメータを指定してください。その他のパラメータは省略可能です。

- filter — 移行するソースオブジェクトの範囲を定義するために以前に作成したフィルタの名前。
- namePath — 特定のソースオブジェクトへの明示的なパス。
- treePath — 評価レポートのコピーを保存するソースデータベースオブジェクトへの明示的なパス。
- forceLoad — true に設定すると、AWS SCT は移行中にデータベースのメタデータツリーを自動的にロードします。デフォルト値は、「false」です。

次のコード例は、ソース Oozie ファイル内の Applications フォルダからファイルを変換します。

```
Convert
    -treePath: 'ETL.APACHE_OOZIE.Applications'
/
```

SaveOnS3 は、Amazon S3 バケットにステートマシンの定義をアップロードします。このコマンドは treePath パラメータを使用します。このコマンドを実行するには、ステートマシン定義を含むターゲットフォルダをこのパラメータの値として使用します。

以下では、AWS_STEP_FUNCTIONS ターゲットオブジェクトの State machine definitions フォルダを Amazon S3 バケットにアップロードします。 AWS SCT は、前提条件 ステップの AWS サービスプロファイルに保存した Amazon S3 バケットを使用します。

```
SaveOnS3
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machine definitions'
/
```

ConfigureStateMachine コマンドはステートマシンを設定します。このコマンドは最大で 6 つのパラメータを使用します。ターゲットスコープは、必ず以下のリストの最初の 3 つのパラメータのいずれかを使用して定義してください。

- filterName — ターゲットオブジェクトのフィルターの名前。CreateFilter コマンドを使用してフィルターを作成できます。
- treePath — ターゲットオブジェクトへの明示的なパス。
- namePath — 特定のターゲットオブジェクトへの明示的なパス。
- iamRole — ステップマシンへのアクセスを提供する IAM ロールの Amazon リソースネーム (ARN)。このパラメータは必須です。

次のコード例では、*role_name* IAM ロールを使用して AWS_STEP_FUNCTIONS で定義されるステートマシンを設定します。

```
ConfigureStateMachine
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machine definitions'
    -role: 'arn:aws:iam::555555555555:role/role_name'
/
```

ApplyToTarget コマンドは、変換されたコードをターゲットサーバーに適用します。このコマンドを実行するには、filterName、treePath、または namePath のいずれかのパラメータを使用して、適用するターゲットオブジェクトを定義します。

次のコード例では、app_wp ステートマシンを AWS Step Functions に適用しています。

```
ApplyToTarget
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machines.app_wp'
/
```

変換したコードがソースコードと同じ結果になるようにするには、AWS SCT 拡張パックを使用できます。これは AWS Step Functions でサポートされていない Apache Oozie 関数をエミュレートする一連の AWS Lambda 関数です。`CreateLambdaExtPack` コマンドを使用して、この拡張パックをインストールできます。

このコマンドは最大 5 つのパラメータを使用します。必ず `extPackId` 用の **`Oozie2SF`** を使用してください。この場合、AWS SCT は Apache Oozie のソース関数用の拡張パックを作成します。

- `extPackId` — 一連の Lambda 関数の一意の識別子。このパラメータは必須です。
- `tempDirectory` — AWS SCT 一時ファイルを保存できるパス。このパラメータは必須です。
- `awsProfile` — AWS プロフィールの名前。
- `lambdaExecRoles` — Lambda 関数に使用する実行ロールの Amazon リソースネーム (ARN) のリスト。
- `createInvokeRoleFlag` — AWS Step Functions の実行ロールを作成するかどうかを示すブール型フラグ。

拡張パックをインストールして使用するには、必要な許可を与えていることを確認します。詳細については、「拡張パック内の AWS Lambda 関数を使用するための権限」を参照してください。

`Convert`、`SaveOnS3`、`ConfigureStateMachine`、`ApplyToTarget`、`CreateLambdaExtPack` コマンドの使用の詳細については、『AWS Schema Conversion Tool CLI コマンドリファレンス』を参照してください。

CLI スクリプトの実行

AWS SCT CLI スクリプトの編集が終了したら、`.scts` 拡張子の付いたファイルとして保存します。これで、AWS SCT インストールパスの `app` フォルダからスクリプトを実行できます。そのためには、次のコマンドを使用します。

```
RunSCTBatch.cmd --pathtoscts "C:\script_path\oozie.scts"
```

前の例では、*`script_path`* を CLI スクリプトを含むファイルへのパスに置き換えます。AWS SCT でのスクリプトの実行の詳細については、「スクリプトモード」を参照してください。

AWS SCT が AWS Step Functions に変換できる Apache Oozie ノード

AWS SCT を使用して Apache Oozie アクションノードと制御フローノードを AWS Step Functions に変換できます。

サポートされているアクションノードは次のとおりです。

- Hive アクション
- Hive2 アクション
- Spark アクション
- MapReduce ストリーミングアクション
- Java アクション
- DistCp アクション
- Pig アクション
- Sqoop アクション
- FS アクション
- Shell アクション

サポートされている制御フローノードは次のとおりです。

- Start アクション
- End アクション
- Kill アクション
- Decision アクション
- Fork アクション
- Join アクション

AWS SCT の AWS DMS との併用

AWS DMS での AWS SCT レプリケーションエージェントの使用

非常に大規模なデータベース移行の場合、AWS SCT レプリケーションエージェント (aws-schema-conversion-tool-dms-agent) を使用して、オンプレミスのデータベースから、Amazon S3 または AWS Snowball Edge デバイスにデータをコピーできます。レプリケーションエージェントは AWS DMS と連動して動作し、AWS SCT がクローズしている間もバックグラウンドで使用できます。

AWS Snowball Edge を使用する場合、AWS SCT エージェントは AWS Snowball デバイスにデータをレプリケートします。次に、デバイスが AWS に送信され、Amazon S3 バケットにデータがロードされます。この期間中、AWS SCT エージェントは引き続き実行されます。次に、エージェントは Amazon S3 のデータを受け取り、ターゲットエンドポイントにコピーします。

詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。

AWS DMS で AWS SCT データ抽出エージェントを使用する

AWS SCT には、Apache Cassandra から Amazon DynamoDB への移行が容易になるデータ抽出エージェント (`aws-schema-conversion-tool-extractor`) があります。Cassandra と DynamoDB は NoSQL データベースですが、システムアーキテクチャとデータ表現が異なります。Cassandra から DynamoDB への移行プロセスの自動化には、AWS SCT のウィザードベースのワークフローを使用します。AWS SCT は AWS Database Migration Service (AWS DMS) と統合して実際のマイグレーションを実行します。

詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。

AWS DMS で AWS SCT を使用しながらログレベルを上げる

例えば、AWS Support で作業する必要がある場合など、AWS DMS で AWS SCT を使用する際に、ログレベルを上げることができます。

AWS SCT をインストールし、必要なドライバをインストールした後、AWS SCT アイコンを選択してアプリケーションを開きます。更新通知が表示された場合は、プロジェクトの完了前または完了後

に更新するかを選択できます。自動プロジェクトウィンドウが開いた場合は、ウィンドウを閉じて、プロジェクトをマニュアルで作成します。

AWS DMS で AWS SCT を使用するときにログレベルを上げるには

1. [Settings] (設定) メニューを開き、[Global Settings] (グローバル設定) を選択します。
2. [Global settings] (グローバル設定) ウィンドウで、[Logging] (ログ) を選択します。
3. デバッグモードでは、[True] を選択します。
4. [Message level] (メッセージレベル) から、次のタイプのログを変更できます。

   - 全般
   - ローダー
   - パーサー
   - プリンタ
   - リゾルバー
   - Telemetry
   - コンバータ

   デフォルトで、すべてのメッセージレベルは [Info] (情報) になっています。
5. 変更するメッセージレベルタイプのログレベルを選択します。

   - トレース (最大限に詳細なログ)
   - デバッグ
   - Info
   - 警告
   - エラー (最小限に詳細なログ)
   - 重大
   - 必須
6. [Apply] (適用) を選択して、プロジェクトの設定を変更します。
7. [OK] を選択し、[Global Settings] (グローバル設定) ダイアログボックスを閉じます。

オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する

AWS SCT エージェントを使用してオンプレミスのデータウェアハウスからデータを抽出し、Amazon Redshift に移行できます。エージェントはデータを抽出し、そのデータを Amazon S3 にアップロードするか、大規模な移行の場合は Edge デバイスにアップロードします。 AWS Snowball その後、 AWS SCT エージェントを使用してデータを Amazon Redshift にコピーできます。

または、 AWS Database Migration Service (AWS DMS) を使用して Amazon Redshift にデータを移行することもできます。 AWS DMS の利点は、継続的なレプリケーション (変更データキャプチャ) をサポートできることです。ただし、データ移行の速度を上げるには、 AWS SCT 複数のエージェントをparallel 使用してください。テストによると、 AWS SCT AWS DMS エージェントはデータ移行を 15 ~ 35% 速く行っています。速度の違いは、データ圧縮、テーブルパーティションの並列移行のサポート、およびさまざまな構成設定によるものです。詳細については、「AWS Database Migration Serviceのターゲットとしての Amazon Redshift データベースの使用」を参照してください。

Amazon S3 はストレージおよび取得サービスです。Amazon S3 にオブジェクトを保存するには、Amazon S3 バケットに保存するファイルをアップロードします。ファイルをアップロードする際に、オブジェクトだけでなく、いずれのメタデータにも権限を設定することができます。

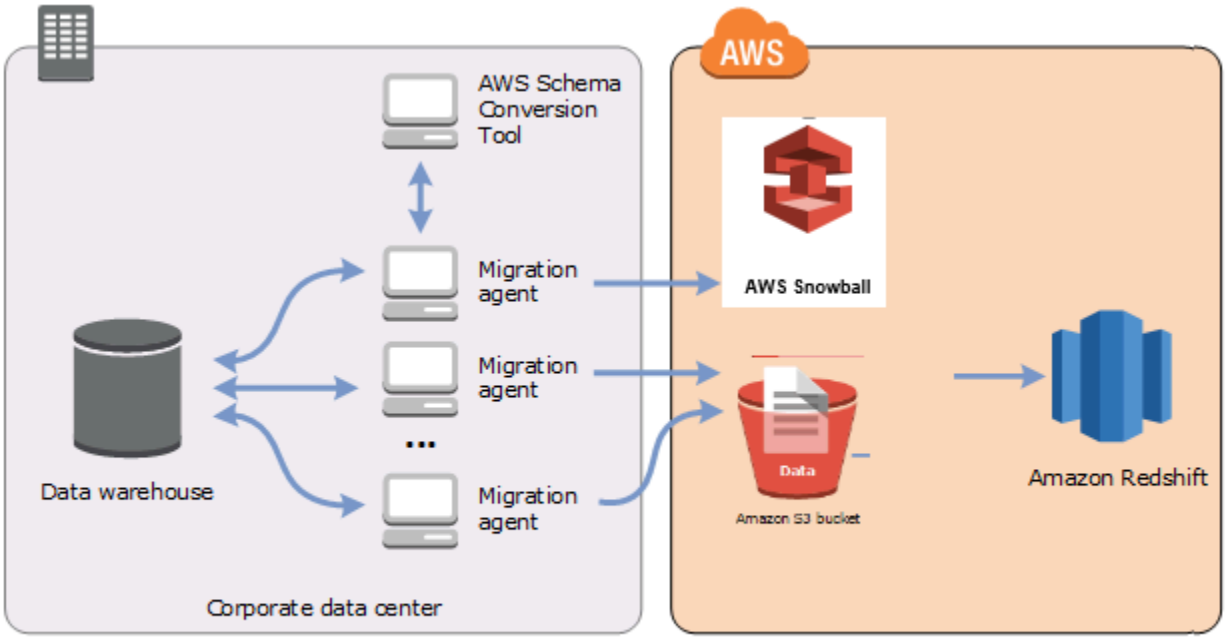
[大規模な移行]

大規模なデータ移行には何テラバイトもの情報が含まれることがあり、ネットワークのパフォーマンスや移動するデータ量の多さによって速度が低下する可能性があります。 AWS Snowball Edge は、 AWS faster-than-network 所有しているアプライアンスを使用してデータを高速にクラウドに転送できるサービスです。 AWS AWS Snowball Edge デバイスには最大 100 TB のデータを保存できます。256 ビットの暗号化と業界標準のトラステッドプラットフォームモジュール (TPM) を使用して、データのセキュリティと完全性の両方を確保します。 chain-of-custody AWS SCT エッジデバイスで動作します。 AWS Snowball

AWS Snowball Edge デバイスを使用する場合 AWS SCT 、データを 2 段階で移行します。まず、AWS SCT を使用してデータをローカルで処理し、 AWS Snowball 次にそのデータをエッジデバイスに移動します。次に、 AWS Snowball Edge AWS プロセスを使用してデバイスを送信し、データを Amazon S3 AWS バケットに自動的にロードします。次に、データが Amazon S3 で使用可能に

なったら、 AWS SCT を使用してデータを Amazon Redshift に移行します。データ抽出エージェントは、 AWS SCT データが閉じている間もバックグラウンドで動作できます。

次の図は、サポートされるシナリオを示しています。



データ抽出エージェントは、現在、以下のソースデータウェアハウスでサポートされています。

- Azure Synapse Analytics
- BigQuery
- Greenplum データベース (バージョン 4.3)
- Microsoft SQL Server (バージョン 2008 以降)
- Netezza (バージョン 7.0.3 以降)
- Oracle (バージョン 10 以降)
- Snowflake (バージョン 3)
- Teradata (バージョン 13 以降)
- Vertica (バージョン 7.2.2 以降)

連邦情報処理標準 (FIPS) のセキュリティ要件に準拠する必要がある場合は、Amazon Redshift の FIPS エンドポイントに接続できます。FIPS AWS エンドポイントは以下の地域で利用できます。

- 米国東部 (バージニア北部) リージョン (redshift-fips.us-east-1.amazonaws.com)
- 米国東部 (オハイオ) リージョン (redshift-fips.us-east-2.amazonaws.com)
- 米国西部 (北カリフォルニア) リージョン (redshift-fips.us-west-1.amazonaws.com)
- 米国西部 (オレゴン) リージョン (redshift-fips.us-west-2.amazonaws.com)

データ抽出エージェントの使用方法については、以下のトピックを参照してください。

トピック

データ抽出エージェントを使用するための前提条件

データ抽出エージェントを使用する前に、Amazon Redshift ユーザーへのターゲットとして Amazon Redshift に必要なアクセス許可を追加します。詳細については、「ターゲットとしての Amazon Redshift の許可」を参照してください。

次に、Amazon S3 バケット情報を保存し、Secure Sockets Layer (SSL) 信頼ストアとキーストアを設定します。

Amazon S3 設定

エージェントは、データを抽出すると Amazon S3 バケットにアップロードします。続行する前に、AWS アカウントと Amazon S3 バケットに接続するための認証情報を入力する必要があります。認証情報とバケット情報をグローバルアプリケーション設定のプロファイルに保存し、 AWS SCT そのプロファイルをプロジェクトに関連付けます。必要に応じて、[グローバル設定] を選択して、新しいプロファイルを作成します。詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。

ターゲット Amazon Redshift データベースにデータを移行するには、 AWS SCT データ抽出エージェントがユーザーに代わって Amazon S3 バケットにアクセスする権限が必要です。この権限を付与するには、以下のポリシーを使用して AWS Identity and Access Management (IAM) ユーザーを作成します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:PutObjectTagging"
            ],
            "Resource": [
                "arn:aws:s3:::bucket_name/*",
                "arn:aws:s3:::bucket_name"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::bucket_name"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Action": [
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:user/DataExtractionAgentName"
            ],
            "Effect": "Allow"
        }
    ]
}
```

前の例では、*bucket_name* を Amazon S3 バケットの名前に置き換えます。次に、*111122223333:user/DataExtractionAgentName* を IAM ユーザーの名前に置き換えます。

IAM ロールの継承

セキュリティを強化するために、 AWS Identity and Access Management (IAM) ロールを使用して Amazon S3 バケットにアクセスできます。そのためには、アクセス許可のないデータ抽出エージェント用の IAM ユーザーを作成します。次に、Amazon S3 アクセスを有効にする IAM ロールを作成し、このロールを引き受けることができるサービスとユーザーのリストを指定します。詳細については、「IAM ユーザーガイド」の「IAM ロール」を参照してください。

Amazon S3 バケットにアクセスするための IAM ロールを設定するには

1. 新しい IAM ユーザーを作成します。ユーザー認証情報には、[プログラムによるアクセス] タイプを選択します。
2. データ抽出エージェントが、 AWS SCT 提供するロールを引き受けることができるようにホスト環境を設定します。前のステップで設定したユーザーが、データ抽出エージェントが認証情報プロバイダーチェーンを使用できるようにしていることを確認してください。詳細については、『AWS SDK for Java デベロッパーガイド』の「認証情報の使用」を参照してください。
3. S3 バケットへのアクセス許可を持つ新しい IAM ロールを作成します。
4. このロールの信頼セクションを変更して、ロールを引き継ぐ前に作成したユーザーを信頼するようにします。次の例では、*111122223333:user/DataExtractionAgentName* をユーザーの名前に置き換えます。

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/DataExtractionAgentName"
    },
    "Action": "sts:AssumeRole"
}
```

5. このロールの信頼セクションを redshift.amazonaws.com を信頼に変更してロールを引き受けます。

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "redshift.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole"
}
```

6. このロールを Amazon Redshift クラスターにアタッチします。

これで、データ抽出エージェントを AWS SCTで実行できます。

IAM ロール割り当てを使用すると、データ移行は次のように機能します。データ抽出エージェントが起動し、認証情報プロバイダーチェーンを使用してユーザー認証情報を取得します。次に、でデータ移行タスクを作成し AWS SCT、データ抽出エージェントが引き受ける IAM ロールを指定して、タスクを開始します。 AWS Security Token Service (AWS STS) は Amazon S3 にアクセスするための一時的な認証情報を生成します。データ抽出エージェントは、これらの認証情報を使用して Amazon S3 にデータをアップロードします。

次に、Amazon Redshift に IAM AWS SCT ロールを提供します。次に、Amazon Redshift AWS STS はから新しい一時的な認証情報を取得して Amazon S3 にアクセスします。Amazon Redshift はこれらの認証情報を使用して、データを Amazon S3 から Amazon Redshift テーブルにコピーします。

セキュリティ設定

AWS Schema Conversion Tool と抽出エージェントはセキュアソケットレイヤー (SSL) を介して通信できます。SSL を有効にするには、信頼ストアとキーストアを設定します。

抽出エージェントとの安全な通信を設定するには

1. を起動します。 AWS Schema Conversion Tool
2. [設定] メニューを開き、[グローバル設定] を選択します。[グローバル設定] ダイアログボックスが表示されます。
3. セキュリティを選択します。
4. [信頼とキーストアを生成] または [既存の信頼ストアとキーストアを選択] を選択します。

   [信頼とキーストアを生成] を選択した場合、信頼ストアとキーストアの名前とパスワードおよび生成されたファイルの場所へのパスを指定します。これらのファイルは、後のステップで使用します。

   [既存の信頼ストアとキーストアを選択] を選択した場合、信頼ストアおよびキーストアのパスワードとファイル名を指定します。これらのファイルは、後のステップで使用します。
5. 信頼ストアおよびキーストアを指定した後で、[OK] を選択し、[グローバル設定] ダイアログボックスを閉じます。

データ抽出エージェントの環境の設定

1 つのホストに複数のデータ抽出エージェントをインストールできます。ただし、1 つのホストで 1 つのデータ抽出エージェントを実行することをお勧めします。

データ抽出エージェントを実行するには、必ず 4 つ以上の vCPUs と 32 GB のメモリを搭載したホストを使用してください。また、使用可能な最小メモリを 4 GB AWS SCT 以上に設定します。詳細については、「追加メモリの構成」を参照してください。

最適な構成とエージェントホストの数は、各顧客の具体的な状況によって異なります。移行するデータ量、ネットワーク帯域幅、データを抽出する時間などの要素を考慮してください。最初に概念実証 (PoC) を実行してから、この PoC の結果に従ってデータ抽出エージェントとホストを構成できます。

抽出エージェントをインストールする

AWS Schema Conversion Toolを実行しているコンピュータとは別の、個別のコンピュータに複数の抽出エージェントをインストールすることをお勧めします。

抽出エージェントは現在、次のオペレーティングシステムでサポートされています。

- Microsoft Windows
- Red Hat Enterprise Linux (RHEL) 6.0
- Ubuntu Linux (バージョン 14.04 以降)

以下の手順を使用して、抽出エージェントをインストールします。抽出エージェントをインストールするそれぞれのコンピュータでこの手順を繰り返します。

抽出エージェントをインストールするには

1. AWS SCT インストーラーファイルをまだダウンロードしていない場合は、のインストール、検証、更新 AWS SCTの指示に従ってダウンロードしてください。 AWS SCT インストーラーファイルを含む.zip ファイルには、抽出エージェントのインストーラーファイルも含まれています。
2. Amazon Corretto 11 の最新バージョンをダウンロードしてインストールします。ダウンロードリンクなどの詳細については、『Amazon Corretto 11 ユーザーガイド』の「Amazon Corretto 11 のダウンロード」を参照してください。
3. agents というサブフォルダ内で移行エージェントのインストーラファイルを見つけます。コンピュータのオペレーティングシステムごとに、抽出エージェントをインストールするための正しいファイルは以下のとおりです。

オペレーティングシステム	ファイル名
Microsoft Windows	`aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.msi`
RHEL	`aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.x86_64.rpm`
Ubuntu Linux	`aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.deb`

4. 新しいコンピュータにインストーラファイルをコピーして、個別のコンピュータで抽出エージェントをインストールします。
5. インストーラファイルを実行します。次に示すように、オペレーティングシステムの手順を使用します。

オペレーティングシステム	インストール手順
Microsoft Windows	ファイルをダブルクリックしてインストーラを実行します。

RHEL	ファイルをダウンロードまたは移動したフォルダーで、次のコマンドを実行します。
	`sudo rpm -ivh aws-schema-conversion-tool-extractor -2.0.1. build-number .x86_64.rpm`
	`sudo ./sct-extractor-setup.sh --config`

Ubuntu Linux	ファイルをダウンロードまたは移動したフォルダーで、次のコマンドを実行します。
	`sudo dpkg -i aws-schema-conversion-tool-extractor -2.0.1. build-number .deb`
	`sudo ./sct-extractor-setup.sh --config`

6. [次へ] を選択し、ライセンス契約に同意して [次へ] を選択します。
7. AWS SCT データ抽出エージェントをインストールするパスを入力し、[次へ] を選択します。
8. [インストール] を選択して、データ抽出エージェントをインストールします。

   AWS SCT データ抽出エージェントをインストールします。インストールを完了するには、データ抽出エージェントを設定します。 AWS SCT 構成セットアッププログラムを自動的に起動します。詳細については、「抽出エージェントを設定する」を参照してください。
9. データ抽出エージェントを設定したら、[完了] を選択してインストールウィザードを閉じます。

抽出エージェントを設定する

以下の手順を使用して、抽出エージェントを設定します。抽出エージェントがインストールされている各コンピュータに対してこの手順を繰り返します。

抽出エージェントを設定するには

1. 設定のセットアッププログラムを起動します。

- Windows では、 AWS SCT データ抽出エージェントのインストール中に構成セットアッププログラムを自動的に起動します。

  必要に応じて、セットアッププログラムを手動で起動できます。そうするには、Windows で `ConfigAgent.bat` ファイルを実行します。このファイルは、エージェントをインストールしたフォルダにあります。

- RHEL および Ubuntu の場合、エージェントをインストールした場所で `sct-extractor-setup.sh` ファイルを実行します。

セットアッププログラムで情報の入力が求められます。それぞれのプロンプトで、デフォルト値が表示されます。

2. 各プロンプトでデフォルト値をそのまま使用するか、新しい値を入力します。

   以下の情報を指定します。

   - [リスニングポート] で、エージェントがリッスンするポート番号を入力します。
   - [ソースベンダーの追加] に [はい] と入力し、ソースデータウェアハウスプラットフォームを入力します。
   - [JDBC ドライバー] で、DBC ドライバーをインストールした場所を入力します。
   - [作業フォルダー] には、 AWS SCT データ抽出エージェントが抽出したデータを保存するパスを入力します。作業フォルダは、エージェントからの他のコンピュータにある場合があります。また、単一の作業フォルダを別々のコンピュータの複数のエージェントで共有することもできます。
   - [SSL 通信を有効にする] には [はい] と入力します。
   - [キーストア] には、キーストアファイルの場所を入力します。
   - [キーストアのパスワード] にキーストアのパスワードを入力します。
   - [クライアント SSL 認証を有効にする] に [はい] と入力します。
   - [信頼ストア] には、信頼ストアファイルの場所を入力します。
   - [信頼ストアのパスワード] に信頼ストアのパスワードを入力します。

セットアッププログラムは抽出エージェント用の設定ファイルを更新します。設定ファイルは `settings.properties` という名前で、抽出エージェントをインストールした場所にあります。

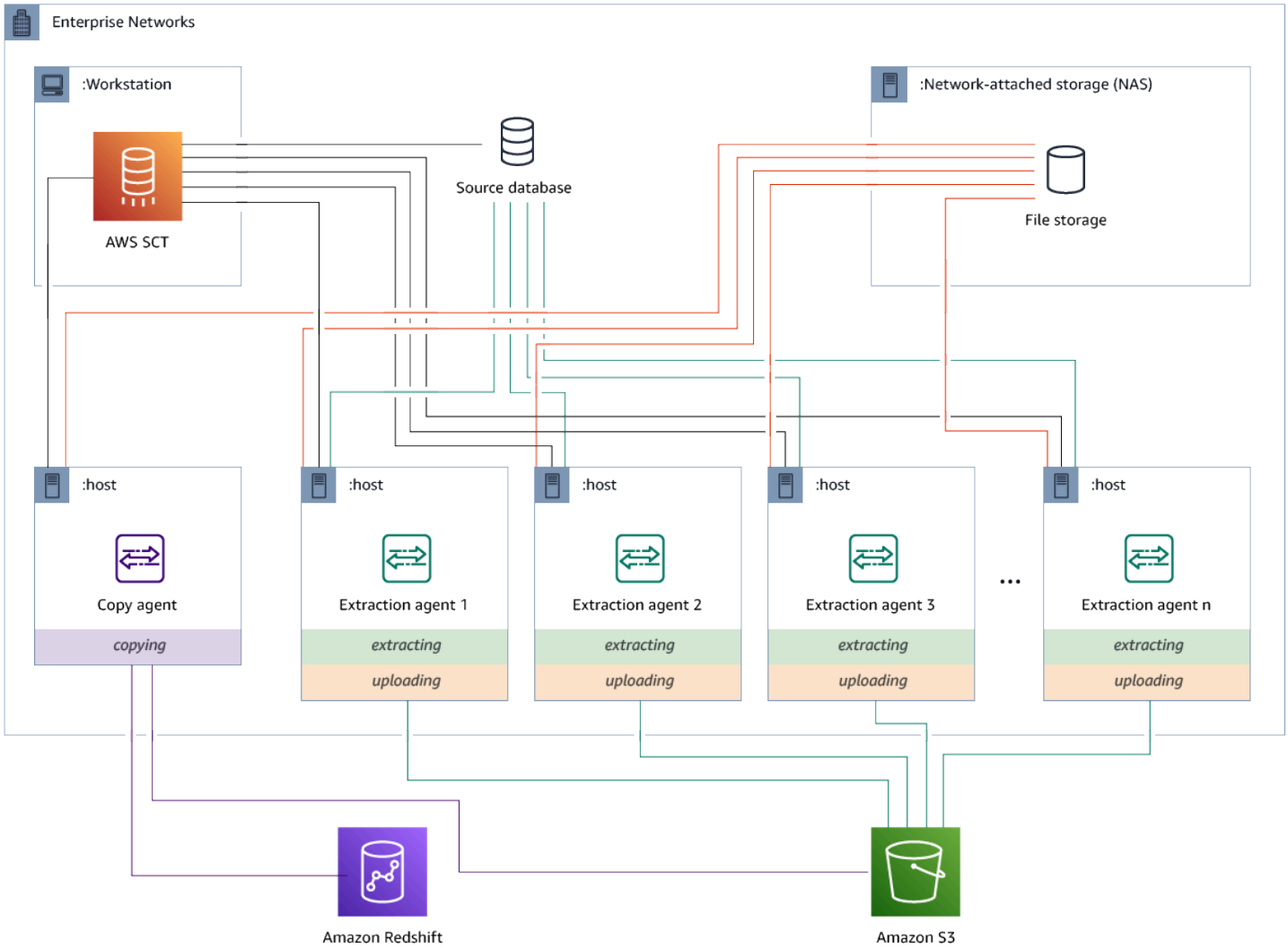
以下に示すのは、サンプルの設定ファイルです。

```
$ cat settings.properties
#extractor.start.fetch.size=20000
#extractor.out.file.size=10485760
#extractor.source.connection.pool.size=20
#extractor.source.connection.pool.min.evictable.idle.time.millis=30000
#extractor.extracting.thread.pool.size=10
vendor=TERADATA
driver.jars=/usr/share/lib/jdbc/terajdbc4.jar
port=8192
redshift.driver.jars=/usr/share/lib/jdbc/RedshiftJDBC42-1.2.43.1067.jar
working.folder=/data/sct
extractor.private.folder=/home/ubuntu
ssl.option=OFF
```

設定を変更するには、テキストエディタを使用して settings.properties ファイルを編集するか、エージェント設定を再実行します。

専用のコピーエージェントを使用した抽出エージェントのインストールと構成

抽出エージェントは、共有ストレージと専用のコピーエージェントのある構成にインストールできます。このシナリオを以下に図表で示します。



この構成は、ソースデータベースサーバーが最大 120 の接続をサポートし、ネットワークに十分なストレージが接続されている場合に有効です。以下の手順を使用して、専用のコピーエージェントのある抽出エージェントを構成します。

抽出エージェントと専用のコピーエージェントをインストールして構成するには

1. すべての抽出エージェントの作業ディレクトリで共有ストレージ上の同じフォルダが使用されていることを確認します。
2. 抽出エージェントをインストールする の手順に従って、エクストラクタエージェントをインストールします。
3. 抽出エージェントを設定する の手順に従って、抽出エージェントを構成しまが、ソース JDBC ドライバのみを指定します。

4. 抽出エージェントを設定する の手順に従って、専用のコピーエージェントを構成しますが、Amazon Redshift JDBC ドライバーのみを指定します。

抽出エージェントを開始する

以下の手順を使用して、抽出エージェントを開始します。抽出エージェントがインストールされている各コンピュータに対してこの手順を繰り返します。

抽出エージェントはリスナーとして機能します。この手順でエージェントを開始すると、エージェントは手順をリッスンし始めます。後のセクションで、データウェアハウスからデータを抽出する手順をエージェントに送信します。

抽出エージェントを開始するには

- 抽出エージェントがインストールされているコンピュータで、次に示す、お使いのオペレーティングシステム用のコマンドを実行します。

オペレーティングシステム	開始コマンド
Microsoft Windows	`StartAgent.bat` バッチファイルをダブルクリックします。
RHEL	エージェントをインストールしたフォルダへのパスで以下のコマンドを実行します。 `sudo initctl start sct-extractor`
Ubuntu Linux	エージェントをインストールしたフォルダへのパスで以下のコマンドを実行します。Ubuntu のバージョンに適切なコマンドを使用します。 Ubuntu 14.04: `sudo initctl start sct-extractor` Ubuntu 15.04 およびそれ以降: `sudo systemctl start sct-extractor`

エージェントの状態を確認するには、同じコマンドを実行します。ただし、`start` を `status` に置き換えます。

エージェントを停止するには、同じコマンドを実行します。ただし、`start` を `stop` に置き換えます。

抽出エージェントをに登録する AWS Schema Conversion Tool

抽出エージェントはを使用して管理します AWS SCT。抽出エージェントはリスナーとして機能します。から指示を受け取ると AWS SCT、データウェアハウスからデータを抽出します。

以下の手順に従って、 AWS SCT 抽出エージェントをプロジェクトに登録します。

抽出エージェントを登録するには

1. を起動し AWS Schema Conversion Tool、プロジェクトを開きます。
2. [ビュー] メニューを開き、[データ移行ビュー] を選択します。[エージェント] タブが表示されます。以前にエージェントを登録したことがある場合は、 AWS SCT で、そのエージェントがタブ上部のグリッドに表示されます。
3. [登録] を選択します。

   AWS SCT エージェントをプロジェクトに登録すると、同じエージェントを別のプロジェクトに登録することはできません。 AWS SCT プロジェクトでエージェントを使用しなくなった場合は、登録を解除できます。登録解除後に、別のプロジェクトに登録できます。
4. [Redshift データエージェント] を選択し、「OK」を選択します。
5. ダイアログボックスの [接続] タブに情報を入力します。

   a. [説明] に、エージェントの説明を入力します。

   b. [ホスト名] に、エージェントのコンピューターのホスト名または IP アドレスを入力します。

   c. [ポート] で、エージェントがリッスンするポート番号を入力します。

   d. [Register] を選択して、 AWS SCT エージェントをプロジェクトに登録します。
6. 前述のステップを繰り返して、 AWS SCT プロジェクトに複数のエージェントを登録します。

エージェントの情報の非表示と回復 AWS SCT

AWS SCT エージェントは、ユーザキートラストストアへのパスワード、データベースアカウント、AWS アカウント情報などの大量の情報を暗号化します。これは、`seed.dat` と呼ばれる特別なファ

イルを使用して実行されます。デフォルトでは、エージェントはこのエージェントを最初に設定するユーザーの作業フォルダで、このファイルを作成します。

さまざまなユーザーがエージェントを設定および実行できるため、 へのパスは seed.dat{extractor.private.folder} ファイルの settings.properties パラメータに保存されます。エージェントが開始すると、このパスを使用して、作用するデータベース用のキー信頼ストア情報にアクセスするための seed.dat ファイルを見つけます。

以下の場合に、エージェントが保存したパスワードの回復が必要になることがあります。

- seed.datユーザーがファイルを紛失しても、 AWS SCT エージェントの場所とポートは変更されていない場合。
- seed.datユーザーがファイルを紛失し、 AWS SCT エージェントの場所とポートが変更された場合。この場合、通常は変更が発生します。これは、エージェントが別のホストまたはポートに移行され、seed.dat ファイル内の情報が無効になっているためです。

これらの場合、エージェントが SSL を使用しないで開始されたときは、エージェントは開始され、以前に作成したエージェントストレージにアクセスできます。次に、復元の待機状態になります。

ただし、これらの場合、エージェントが SSL を使用して開始されたときは、エージェントを再起動することはできません。これは、エージェントが settings.properties ファイルに保存された証明書のパスワードを復号できないためです。このような種類のスタートアップでは、エージェントは起動に失敗します。次のようなエラーがログに記録されます。「SSL モードを有効にしてエージェントを開始できませんでした。エージェントを再設定してください。理由: キーストアのパスワードが正しくありません。」

この問題を修正するには、新しいエージェントを作成し、SSL 証明書にアクセスするための既存のパスワードを使用するようにエージェントを設定します。そのためには、次の手順を使用します。

この手順を実行すると、エージェントは実行され、「回復待ち」状態になります。 AWS SCT 「回復待ち」状態のエージェントに、必要なパスワードを自動的に送信します。エージェントがパスワードを取得すると、任意のタスクを再開できます。 AWS SCT 側でそれ以上のユーザーアクションは必要ありません。

エージェントを再設定し、SSL 証明書にアクセスするためのパスワードを復元するには

1. AWS SCT 新しいエージェントをインストールし、設定を実行します。

2. agent.name ファイルの instance.properties プロパティを、ストレージが作成されたエージェントの名前に変更し、新しいエージェントが既存のエージェントストレージに対して動作できるようにします。

   instance.properties ファイルは、エージェントのプライベートフォルダに保存されます。このフォルダは次の命名規則を使用して名前が付けられます。{*output.folder*}\dmt\{*hostName*}_{*portNumber*}\。
3. {*output.folder*} の名前を、前のエージェントの出力フォルダの名前に変更します。

   この時点では AWS SCT 、まだ古いホストとポートの古いエクストラクターにアクセスしようとしています。その結果、アクセス不可能なエクストラクタは、FAILED ステータスになります。次に、ホストおよびポートを変更できます。
4. 古いエージェントのホスト、ポート、またはその両方を変更します。そのためには、Modify コマンドを使用して、リクエストのフローを新しいエージェントにリダイレクトします。

新しいエージェントに ping AWS SCT できると、 AWS SCT エージェントから「回復待ち」というステータスが表示されます。 AWS SCT その後、エージェントのパスワードが自動的に回復されます。

エージェントストレージを操作する各エージェントは、storage.lck にある {*output.folder*}\{*agentName*}\storage\ という特別なファイルを更新します。このファイルには、エージェントのネットワーク ID と、ストレージがロックされるまでの時間が含まれています。エージェントがエージェントストレージを操作する場合、storage.lck ファイルを更新し、ストレージのリースを 5 分ごとに 10 分延長します。リースの有効期限が切れるまでは、他のインスタンスはこのエージェントストレージを操作できません。

でのデータ移行ルールの作成 AWS SCT

を使用してデータを抽出する前に AWS Schema Conversion Tool、抽出するデータ量を減らすフィルタを設定できます。WHERE 句を使用して抽出するデータを減らすことで、データ移行ルールを作成できます。例えば、単一のテーブルからデータを選択する WHERE 句を記述できます。

データ移行ルールを作成し、プロジェクトの一部としてフィルタを保存できます。プロジェクトを開いた状態で、以下の手順でデータ移行ルールを作成します。

データ移行ルールを作成するには

1. [ビュー] メニューを開き、[データ移行ビュー] を選択します。

2. [データ移行ルール] を選択し、[新しいルールを追加] を選択します。
3. データ移行ルールを設定します。

   a. [名前] に、データ移行ルールの名前を入力します。
   b. [スキーマ名の候補] に、スキーマに適用するフィルタを入力します。このフィルタで、WHERE 句の評価には LIKE 句が使用されます。1 つのスキーマを選択するには、正確なスキーマ名を入力します。複数のスキーマを選択するには、「%」文字をワイルドカードとして使用して、スキーマ名の任意の数の文字と一致させます。
   c. [テーブル名の候補] に、テーブルに適用するフィルタを入力します。このフィルタで、WHERE 句の評価には LIKE 句が使用されます。1 つのテーブルを選択するには、正確な名前を入力します。複数のテーブルを選択するには、「%」文字をワイルドカードとして使用して、テーブル名の任意の数の文字と一致させます。
   d. [Where 句] に、データをフィルタする WHERE 句を入力します。
4. フィルタを設定したら、[保存] を選択してフィルタを保存します。変更をキャンセルする場合は、[キャンセル] を選択します。
5. ルールの追加、編集、削除が完了したら、[すべてを保存] を選択して変更内容を保存します。

フィルタを削除せずに無効にするには、切り替えアイコンを使用します。既存のフィルタを複製するには、コピーアイコンを使用します。既存のフィルタを削除するには、削除アイコンを使用します。フィルタへの変更を保存するには、[すべてを保存] を選択します。

プロジェクト設定からエクストラクタとコピー設定を変更する

の [プロジェクト設定] ウィンドウから AWS SCT、データ抽出エージェントと Amazon Redshift COPY コマンドの設定を選択できます。

これらの設定を選択するには、[設定]、[プロジェクト設定]、[データ移行] の順に選択します。ここでは、[抽出設定]、[Amazon S3 設定]、および [コピー設定] を編集できます。

次の表の手順を使用して、[抽出設定] の情報を入力します。

このパラメータについて	この操作を行います
圧縮形式	入力ファイルの圧縮形式を指定します。[GZIP]、[BZIP2]、[ZSTD]、または[圧縮なし] のいずれかのオプションを選択します。

このパラメータについて	この操作を行います
[区切り文字]	入力ファイルのフィールドを区切る ASCII 文字を指定します。印刷されない文字はサポートされていません。
[NULL 値を文字列として表す]	データに NULL ターミネータが含まれている場合は、このオプションをオンにしてください。このオプションをオフにすると、Amazon Redshift COPY コマンドは NULL をレコードの最後として扱い、ロードプロセスを終了します。
[ソート戦略]	ソートを使用して、失敗した時点から抽出を再開します。[最初の失敗後に並べ替えを使用する (推奨)]、[可能であれば並べ替えを使用する]、[並べ替えを行わない] のいずれかを選択します。詳細については、「the section called “データをソートする”」を参照してください。
[ソースー時スキーマ]	抽出エージェントがテンポラリオブジェクトを作成できるソースデータベースのスキーマの名前を入力します。
[出力ファイルサイズ (MB)]	Amazon S3 にアップロードするファイルのサイズ (MB) を入力します。
[Snowball Out のファイルサイズ (MB)]	アップロードするファイルのサイズを MB 単位で入力します。AWS Snowballファイルのサイズは 1 ~ 1,000 MB です。
自動パーティショニングを使用します。[Greenplum と Netezza の場合は、サポートされているテーブルの最小サイズ (メガバイト単位) を入力する]	このオプションをオンにしてテーブルパーティショニングを使用し、Greenplum と Netezza のソースデータベースでパーティション化するテーブルのサイズを入力します。Oracle から Amazon Redshift への移行では、 AWS SCT すべてのパーティションテーブルにサブタスクが作成されるため、このフィールドは空のままにしておくことができます。
[LOB を抽出]	このオプションをオンにすると、ソースデータベースからラージオブジェクト (LOB) を抽出できます。LOB には、BLOB、CLOB、NCLOB、XML ファイルなどが含まれます。LOB ごとに、 AWS SCT 抽出エージェントはデータファイルを作成します。

このパラメータについて	この操作を行います
[Amazon S3 バケット LOB フォルダ]	AWS SCT 抽出エージェントが LOB を保存する場所を入力します。
[RTRIM を文字列列に適用する]	このオプションをオンにすると、指定された一連の文字が抽出された文字列の末尾から切り捨てます。
[S3 にアップロードした後、ファイルをローカルに保つ]	このオプションをオンにすると、データ抽出エージェントが Amazon S3 にファイルをアップロードした後もファイルをローカルマシンに保存します。

次の表の手順を使用して、[Amazon S3 設定] の情報を入力します。

このパラメータについて	この操作を行います
[プロキシの使用]	このオプションをオンにすると、プロキシサーバーを使用して Amazon S3 にデータをアップロードします。次に、データ転送プロトコルを選択し、ホスト名、ポート、ユーザー名、パスワードを入力します。
[エンドポイントタイプ]	[FIPS] を選択して連邦情報処理規格 (FIPS) エンドポイントを使用します。[VPCE] を選択して、仮想プライベートクラウド (VPC) エンドポイントを使用します。次に、[VPC エンドポイント] に VPC エンドポイントのドメインネームシステム (DNS) を入力します。
[Amazon Redshift にコピーした後もファイルを Amazon S3 に保存]	このオプションをオンにすると、抽出したファイルを Amazon Redshift にコピーした後も Amazon S3 に残ります。

次の表の指示に従って、[コピー設定] の情報を入力します。

このパラメータについて	この操作を行います
[最大エラー数]	ロードエラーの数を入力します。操作がこの制限に達すると、AWS SCT データ抽出エージェントはデータロードプロセスを

このパラメータについて	この操作を行います
	終了します。デフォルト値は 0 です。つまり、 AWS SCT データ抽出エージェントは障害が発生してもデータのロードを続行します。
[無効な UTF-8 文字を置換]	このオプションをオンにすると、有効でない UTF-8 文字を指定した文字に置き換え、データロード操作を続行します。
[NULL 値に空白を使用]	このオプションをオンにすると、空白文字で構成される空白フィールドが NULL としてロードされます。
[NULL 値に空を使用]	このオプションをオンにすると、空の CHARおよび VARCHAR フィールドが NULL としてロードされます。
[列を切り捨てる]	このオプションをオンにすると、データ型の仕様に合わせて列内のデータが切り捨てられます。
[自動圧縮]	このオプションをオンにすると、コピー操作中に圧縮エンコードが適用されます。
[統計情報の自動更新]	このオプションをオンにすると、コピー操作の最後に統計が更新されます。
[ロード前にファイルをチェック]	このオプションをオンにすると、データファイルを Amazon Redshift にロードする前に検証できます。

を使用して移行前にデータをソートします。 AWS SCT

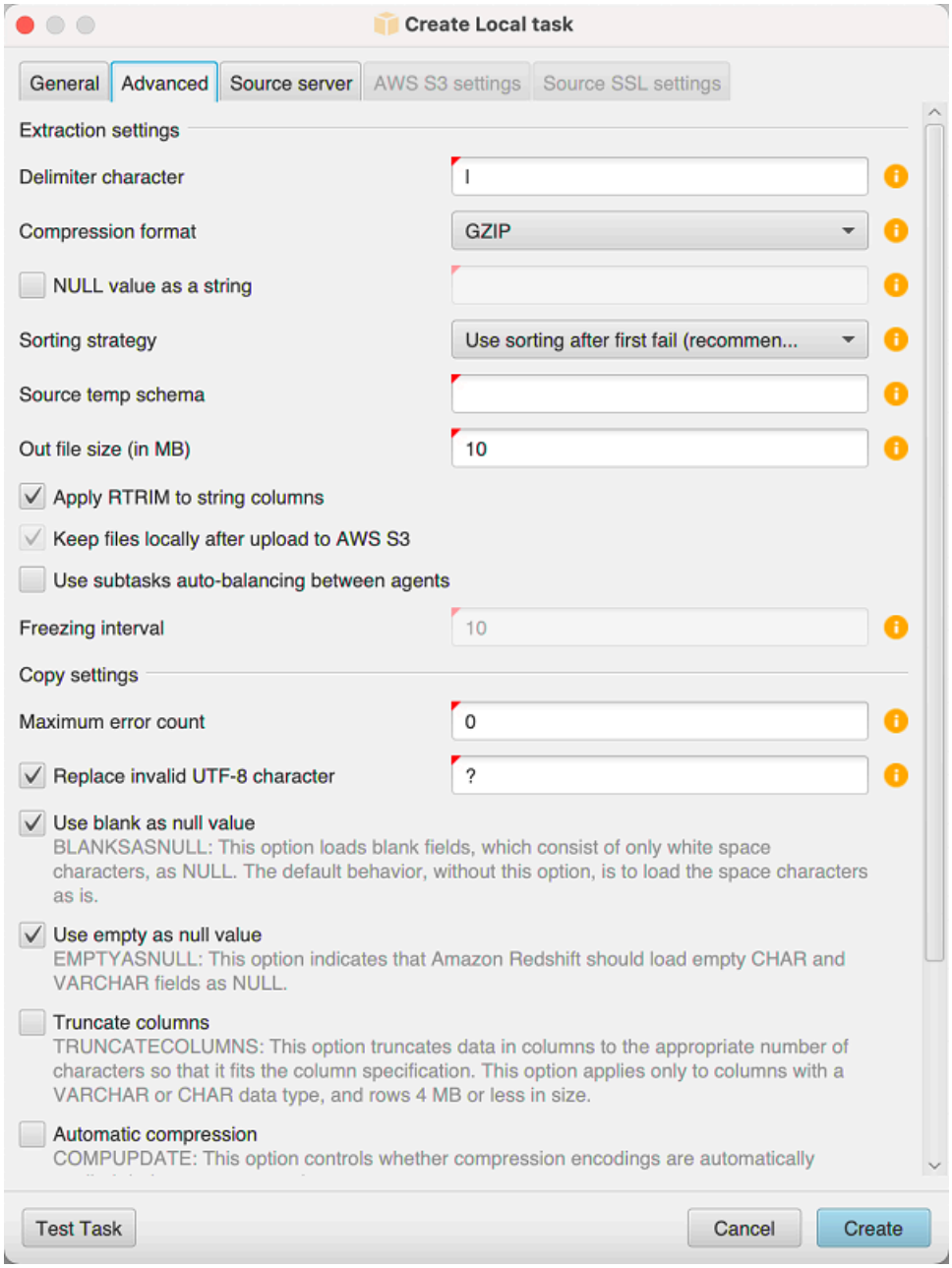
AWS SCT を使用して移行前にデータをソートすることには、いくつかの利点があります。最初にデータをソートすると、 AWS SCT 障害発生後に最後に保存した時点で抽出エージェントを再起動できます。また、Amazon Redshift にデータを移行する場合、最初にデータをソートすると、Amazon Redshift AWS SCT にデータをより速く挿入できます。

これらの利点は、 AWS SCT データ抽出クエリの作成方法に関係しています。場合によっては、これらのクエリで DENSE_RANK AWS SCT 分析関数を使用します。ただし、DENSE_RANK は抽出結果のデータセットをソートするために多くの時間とサーバーリソースを費やすことがあるため、DENSE_RANK AWS SCT がなくても動作する場合は機能します。

移行前にデータを並べ替えるには、以下を使用します。 AWS SCT

1. AWS SCT プロジェクトを開きます。
2. オブジェクトのコンテキスト (右クリック) メニューを開き、[ローカルタスクの作成] を選択します。
3. [アドバンス] タブを選択し、[ソート戦略] で以下のいずれかのオプションを選択します。

   - [Never use sorting] (ソートを使用しない) 抽出エージェントは DENSE_RANK 分析関数を使用せず、障害が発生した場合は最初からやり直します。
   - [Use sorting if possible] (可能な場合はソートを使用) テーブルにプライマリキーまたは一意の制約事項がある場合、抽出エージェントは DENSE_RANK を使用します。
   - [Use sorting after first fail] (最初の失敗の後ソートを使用) (推奨) 抽出エージェントは、まず DENSE_RANK を使用せずにデータを取得しようとします。最初の試行が失敗すると、抽出エージェントは DENSE_RANK を使用してクエリを再構築し、障害発生時の場所を保持します。



4. 以下で説明するように追加のパラメータを設定し、[Create] (作成) を選択してデータ抽出タスクを作成します。

AWS SCT データ抽出タスクの作成、実行、監視

データ抽出タスクを作成、実行、モニタリングするには、以下の手順を使用します。

エージェントにタスクを割り当て、データを移行するには

1. で AWS Schema Conversion Tool、スキーマを変換したら、プロジェクトの左側のパネルから 1 つまたは複数のテーブルを選択します。

   すべてのテーブルを選択することもできますが、パフォーマンス上の理由からそうしないことをお勧めします。データウェアハウスのテーブルのサイズに基づいて、複数のテーブルに複数のタスクを作成することをお勧めします。
2. 各テーブルのコンテキスト (右クリック) メニューを開き、[タスクの作成]を選択します。「ローカルタスクの作成」ダイアログボックスが開きます。
3. [タスク名] にタスクの名前を入力します。
4. [移動モード] で、以下のいずれかを選択します。

   - [抽出のみ] - データを抽出し、ローカル作業フォルダに保存します。
   - [抽出およびアップロード] - データを抽出し、Amazon S3 にアップロードします。
   - [抽出、アップロード、コピー] - データを抽出し、Amazon S3 にアップロードして、Amazon Redshift データウェアハウスにコピーします。
5. [暗号化タイプ] で、以下のいずれかを選択します。

   - [なし] — データ移行プロセス全体でデータ暗号化を無効にします。
   - [CSE_SK] — 対称鍵によるクライアント側の暗号化を使用してデータを移行します。 AWS SCT は暗号化キーを自動的に生成し、Secure Sockets Layer (SSL) を使用してデータ抽出エージェントに送信します。 AWS SCT は、データ移行中にラージオブジェクト (LOB) を暗号化しません。
6. ラージオブジェクトを抽出するには、[Extract LOBs] (LOBを抽出) を選択します。ラージオブジェクトを抽出する必要がない場合は、このチェックボックスをオフにします。オフにすると、抽出するデータの量が減ります。
7. タスクの詳細情報を表示するには、[ログを有効] を選択します。タスクログを使用して問題をデバッグできます。

   タスクのログ記録を有効にする場合は、表示する詳細レベルを選択します。レベルは次のとおりです。各レベルには、前のレベルのすべてのメッセージが含まれます。

   - ERROR - 最小量の詳細。
   - WARNING
   - INFO

- DEBUG
- TRACE - 最大量の詳細。

8. データをエクスポートするには BigQuery、Google Cloud Storage AWS SCT バケットフォルダを使用します。このフォルダには、データ抽出エージェントがソースデータを保存します。

   Google Cloud Storage バケットフォルダへのパスを入力するには、[詳細設定] を選択します。[Google CS バケットフォルダ] には、バケット名とフォルダ名を入力します。

9. データ抽出エージェントユーザーのロールを引き受けるには、[Amazon S3 設定] を選択します。[IAM ロール] に、使用するロールの名前を入力します。[リージョン] では、 AWS リージョン このロールに適したを選択します。

10. [タスクのテスト] を選択して、作業フォルダ、Amazon S3 バケット、および Amazon Redshift データウェアハウスに接続できることを確認します。検証は、選択した移行モードによって異なります。

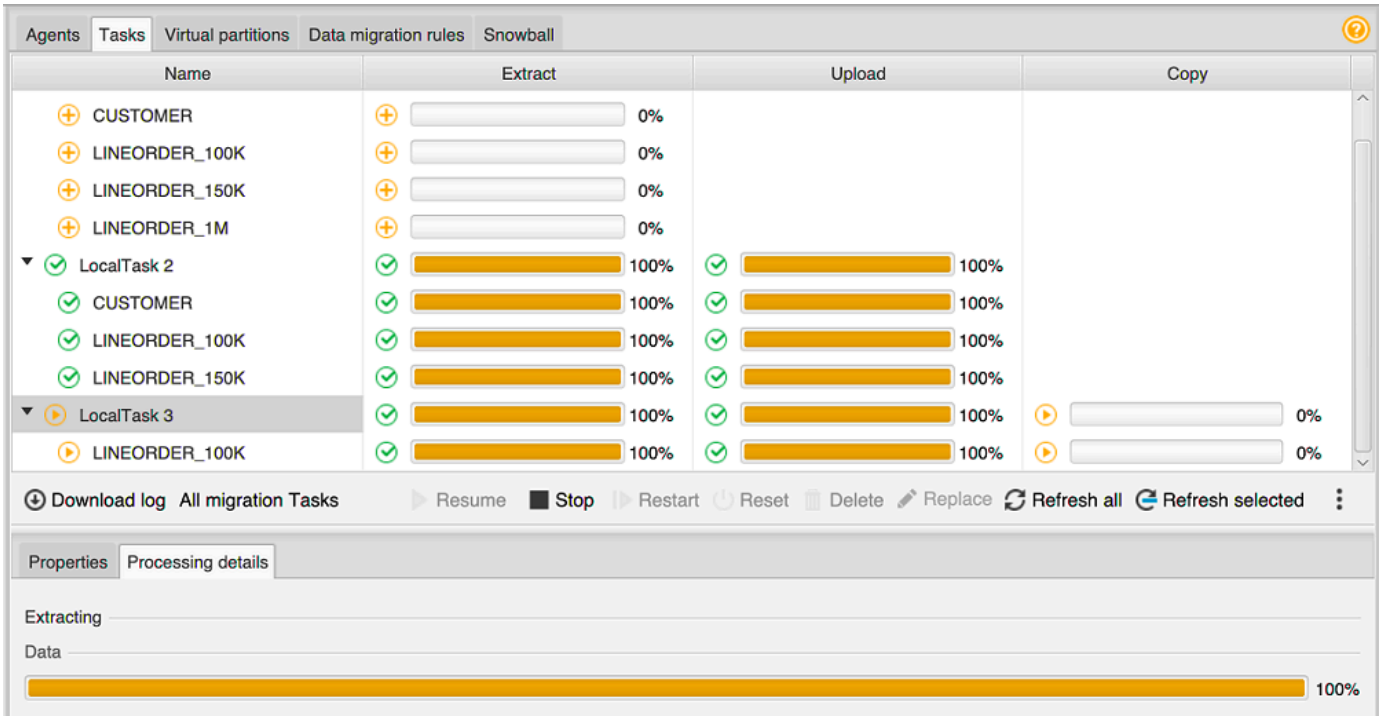
11. 作成を選択して、タスクを作成します。

12. 前述の手順を繰り返して、移行するすべてのデータに対してタスクを作成します。

タスクを実行およびモニタリングするには

1. [ビュー] の [データ移行ビュー] を選択します。[エージェント] タブが表示されます。

2. [タスク] タブを選択します。タスクは、次に示すように上部にあるグリッドに表示されます。タスクのステータスは上部グリッドで確認できます。サブタスクのステータスは下部グリッドで確認できます。



3. 上部グリッドのタスクを選択して展開します。選択した移行モードに応じて、タスクは [Extract] (抽出)、[Upload] (アップロード)、および [Copy] (コピー) に分類されます。
4. タスクの [Start] (開始) を選択してタスクを開始します。作業中に、タスクのステータスをモニタリングできます。サブタスクは並行して実行されます。抽出、アップロード、およびコピーも並行して実行されます。
5. タスクの設定時にログ記録を有効にした場合は、ログを表示できます。

    a. [ログのダウンロード] を選択します。ログファイルを含むフォルダ名を示すメッセージが表示されます。メッセージを閉じます。

    b. [タスクの詳細] タブにリンクが表示されます。リンクを選択すると、ログファイルが含まれたフォルダが開きます。

終了しても AWS SCT、エージェントとタスクは引き続き実行されます。 AWS SCT 後で再び開いて、タスクのステータスを確認したり、タスクログを確認したりできます。

エクスポートとインポートを使用して、データ抽出タスクをローカルディスクに保存し、同じプロジェクトまたは別のプロジェクトに復元することができます。タスクをエクスポートするには、プロジェクト内に抽出タスクが少なくとも 1 つ作成されていることを確認します。1 つの抽出タスク、またはプロジェクトで作成されたすべてのタスクを読み込むことができます。

抽出タスクをエクスポートすると、 AWS SCT `.xml`そのタスク用に別のファイルが作成されます。`.xml` ファイルには、タスクのプロパティ、説明、サブタスクなど、そのタスクのメタデータ情報が保存されます。`.xml` ファイルには、抽出タスクの処理に関する情報は含まれていません。タスクがインポートされると、次のような情報が再作成されます。

- タスクの進行状況
- サブタスクとステージの状態
- サブタスクとステージによる抽出エージェントの分布
- タスク ID とサブタスク ID
- [タスク名]

AWS SCT データ抽出タスクのエクスポートとインポート

AWS SCT エクスポートとインポートを使用して、あるプロジェクトから既存のタスクを簡単に保存し、別のプロジェクト (または同じプロジェクト) に復元できます。次の手順に従って、データ抽出タスクをエクスポートおよびインポートします。

データ抽出タスクをエクスポートおよびインポートするには

1. [ビュー] の [データ移行ビュー] を選択します。[エージェント] タブが表示されます。
2. [Tasks] (タスク) タブを選択します。タスクは表示されるグリッドに一覧表示されます。
3. タスクのリストの右下隅にある 3 つの縦に整列したドット (省略記号アイコン) を選択します。
4. ポップアップメニューから [Export task] (タスクのエクスポート) を選択します。
5. AWS SCT `.xml`タスクエクスポートファイルを保存するフォルダーを選択します。

   AWS SCT というファイル名形式のタスクエクスポートファイルを作成します*TASK-DESCRIPTION_TASK-ID*`.xml`。
6. タスクのリストの右下にある 3 つの縦に整列したドット (省略記号アイコン) を選択します。
7. ポップアップメニューから [Import task] (タスクのインポート) を選択します。

   抽出タスクをソースデータベースに接続されているプロジェクトにインポートでき、プロジェクトには少なくとも 1 つのアクティブな登録済みの抽出エージェントがあります。
8. エクスポートした抽出タスクの `.xml` ファイルを選択します。

   AWS SCT 抽出タスクのパラメータをファイルから取得し、タスクを作成して、抽出エージェントにタスクを追加します。

9. この手順を繰り返して、追加のデータ抽出タスクをエクスポートおよびインポートします。

このプロセスが終了するとエクスポートとインポートが完了し、データ抽出タスクを使用する準備が整います。

AWS Snowball Edge デバイスを使用したデータ抽出

AWS Snowball Edge AWS SCT を使用するプロセスには複数のステップがあります。移行には、AWS SCT データ抽出エージェントを使用してデータをエッジデバイスに移動するローカルタスクと、 AWS Snowball エッジデバイスから Amazon S3 AWS バケットにデータをコピーする中間アクションが含まれます。 AWS Snowball このプロセスにより、Amazon S3 バケットから Amazon Redshift AWS SCT へのデータのロードが完了します。

この概要に続くセクションでは、 step-by-step これらの各タスクについてガイドします。この手順では、 AWS SCT データ抽出エージェントを専用マシンにインストールし、設定して登録していることを前提としています。

AWS Snowball Edge を使用してローカルデータストアからデータストアにデータを移行するには、AWS 次の手順を実行します。

1. AWS Snowball コンソールを使用して AWS Snowball Edge ジョブを作成します。
2. ローカルの専用 Linux マシンを使用して AWS Snowball Edge デバイスのロックを解除します。
3. で新しいプロジェクトを作成します AWS SCT。
4. データ抽出エージェントをインストールして構成します。
5. Amazon S3 バケットが使用するアクセス許可を作成および設定します。
6. AWS Snowball AWS SCT プロジェクトにジョブをインポートします。
7. AWS SCTにデータ抽出エージェントを登録します。
8. でローカルタスクを作成します AWS SCT。
9. AWS SCTでデータ移行タスクを実行して監視します。

S: Edge tep-by-step AWS SCT を使用してデータを移行する手順 AWS Snowball

以下のセクションでは、移行ステップに関する詳細情報を提供します。

ステップ 1: AWS Snowball Edge ジョブを作成する

『Edge 開発者ガイド』の「AWS Snowball エッジジョブの作成」セクションで説明されている手順に従ってジョブを作成します。 AWS Snowball AWS Snowball

ステップ 2: AWS Snowball Edge デバイスのロックを解除する

AWS DMS エージェントをインストールしたマシンから Snowball Edge デバイスのロックを解除して認証情報を提供するコマンドを実行します。これらのコマンドを実行することで、 AWS DMS エージェントコールが AWS Snowball Edge デバイスに接続されていることを確認できます。 AWS Snowball Edge デバイスのロック解除の詳細については、「Snowball Edge のロック解除」を参照してください。

```
aws s3 ls s3://<bucket-name> --profile <Snowball Edge profile> --endpoint http://
<Snowball IP>:8080 --recursive
```

ステップ 3: 新しいプロジェクトを作成する AWS SCT

次に、 AWS SCT 新しいプロジェクトを作成します。

で新規プロジェクトを作成するには AWS SCT

1. を起動します AWS Schema Conversion Tool。[ファイル] メニューで [新しいプロジェクト] を選択します。[新しいプロジェクト] ダイアログボックスが表示されます。
2. コンピュータにローカルに保存されているプロジェクトの名前を入力します。
3. ローカルプロジェクトファイルの場所を入力します。
4. 「OK」 AWS SCT を選択してプロジェクトを作成します。
5. 「Add source」を選択して、 AWS SCT 新しいソースデータベースをプロジェクトに追加します。
6. [ターゲットを追加] を選択して、 AWS SCT 新しいターゲットプラットフォームをプロジェクトに追加します。
7. 左側のパネルでソースデータベーススキーマを選択します。
8. 右側のパネルでは、選択したソーススキーマのターゲットデータベースプラットフォームを指定します。
9. [Create mapping] (マッピングの作成) を選択します。このボタンは、ソースデータベーススキーマとターゲットデータベースプラットフォームを選択するとアクティブになります。

ステップ 4: データ抽出エージェントをインストールして構成する

AWS SCT データ抽出エージェントを使用して Amazon Redshift にデータを移行します。インストール用にダウンロードした.zip ファイルには AWS SCT、抽出エージェントのインストーラファイルが含まれています。データ抽出エージェントは Windows、Red Hat Enterprise Linux、または Ubuntu にインストールできます。詳細については、「抽出エージェントをインストールする」を参照してください。

データ抽出エージェントを設定するには、ソースとターゲットのデータベースエンジンを入力します。また、データ抽出エージェントを実行するコンピューターに、ソースデータベースとターゲットデータベース用の JDBC ドライバーがダウンロードされていることを確認してください。データ抽出エージェントは、これらのドライバーを使用して、移行元のデータベースに接続できます。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。

Windows では、データ抽出エージェントインストーラーがコマンドプロンプトウィンドウで構成ウィザードを起動します。Linux では、エージェントをインストールした場所から `sct-extractor-setup.sh` ファイルを実行します。

ステップ 5: Amazon S3 AWS SCT バケットにアクセスするように設定する

Amazon S3 バケットの作成については、『Amazon Simple Storage Service ユーザーガイド』の「バケットの概要」を参照してください。

ステップ 6: AWS SnowballAWS SCT ジョブをプロジェクトにインポートする

AWS SCT プロジェクトを AWS Snowball Edge デバイスに接続するには、 AWS Snowball ジョブをインポートします。

AWS Snowball ジョブをインポートするには

1. [設定] メニューを開き、[グローバル設定] を選択します。[グローバル設定] ダイアログボックスが表示されます。
2. [AWS サービスプロファイル] を選択し、[ジョブのインポート] を選択します。
3. AWS Snowball ジョブを選択してください。
4. [AWS Snowball IP] を入力します。詳細については、『AWS Snowball ユーザーガイド』の「IP アドレスの変更」を参照してください。
5. AWS Snowball ポートを入力します。詳細については、『Edge 開発者ガイド』の「AWSAWS Snowball エッジデバイスでサービスを使用するために必要なポート」を参照してください。AWS Snowball

6. [AWS Snowball アクセスキー] と [AWS Snowball シークレットキー] を入力します。詳細については、『AWS Snowball ユーザーガイド』の「AWS Snowballでの認証とアクセスコントロール」を参照してください。
7. [適用]、[OK] の順に選択します。

ステップ 7: データ抽出エージェントをに登録する AWS SCT

このセクションでは、 AWS SCTデータ抽出エージェントをに登録します。

データ抽出エージェントを登録するには

1. [ビュー] メニューで [データ移行ビュー (その他)] を選択し、[登録] を選択します。
2. [説明] に、データ抽出エージェントの名前を入力します。
3. [ホスト名] には、データ抽出エージェントを実行するコンピューターの IP アドレスを入力します。
4. [ポート] には、設定したリスニングポートを入力します。
5. [登録] を選択します。

ステップ 8: ローカルタスクを作成する

次に、移行タスクを作成します。このタスクには 2 つのサブタスクが含まれます。1 つのサブタスクでソースデータベースから AWS Snowball Edge アプライアンスにデータを移行します。もう 1 つのサブタスクはアプライアンスが Amazon S3 バケットにロードするデータを受け取り、ターゲットデータベースに移行します。

移行タスクを作成するには

1. [ビュー] メニューを開き、[データ移行ビュー] を選択します。
2. ソースデータベースのスキーマを表示する左のパネルで、移行するスキーマオブジェクトを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[ローカルタスクの作成] を選択します。
3. [タスク名] に、データ移行タスクのわかりやすい名前を入力します。
4. [移行モード] には、[抽出、アップロード、コピー] を選択します。
5. [Amazon S3 設定] を選択します。
6. [Snowball を使用] を選択します。

7. データ抽出エージェントがデータを保存できる Amazon S3 バケットのフォルダとサブフォルダを入力します。
8. 作成を選択して、タスクを作成します。

ステップ 9: でのデータ移行タスクの実行と監視 AWS SCT

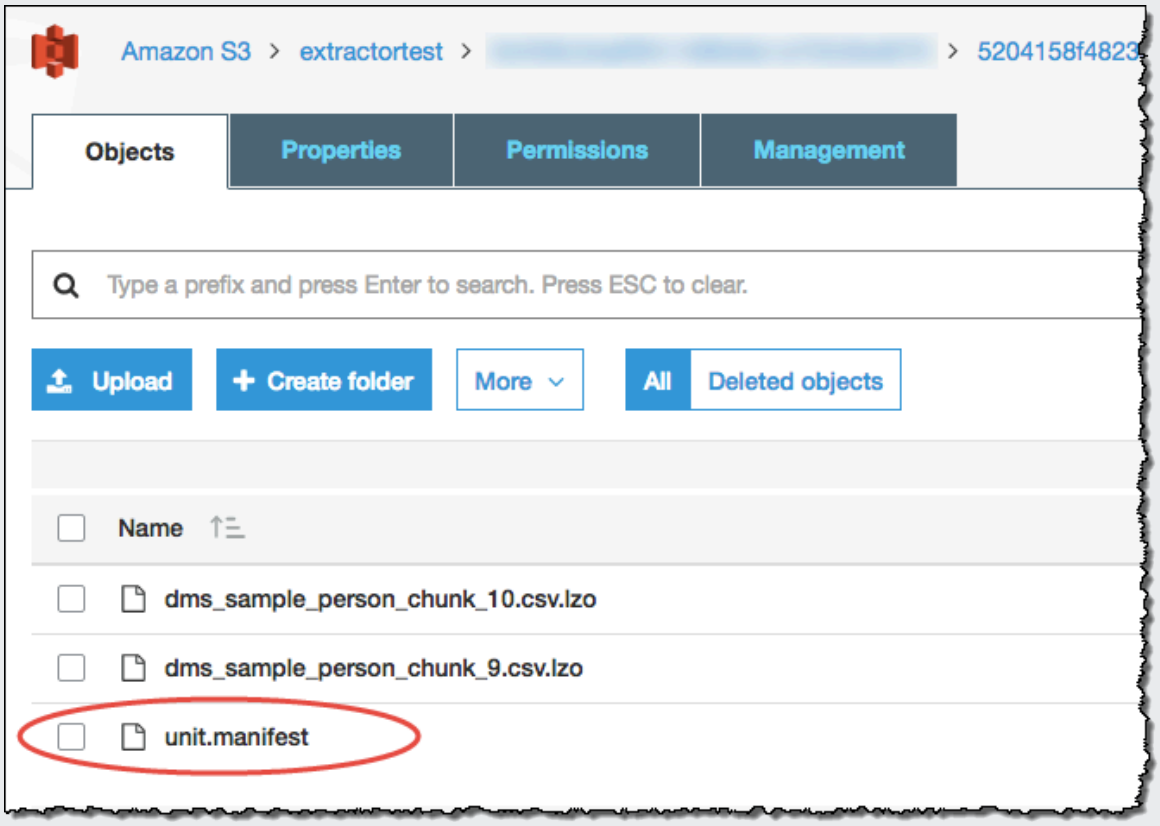
データ移行タスクを開始するには、[開始] を選択します。ソースデータベース、Amazon S3 バケット、 AWS Snowball デバイスへの接続と、上のターゲットデータベースへの接続が確立されていることを確認します AWS。

[タスク] タブでは、データ移行タスクとそのサブタスクを監視および管理できます。データ移行の進行状況を確認できるほか、データ移行タスクを一時停止または再開できます。

データ抽出タスクの出力

移行タスクが完了すると、データは準備ができています。次の情報を使用して、選択した移行モードと、データの場所に応じて処理方法を決定します。

移行モード	データの場所
[抽出、アップロード、およびコピー]	データはすでに Amazon Redshift データウェアハウスに存在しています。データがそこにあることを確認し、使用を開始できます。詳細については、「クライアントツールおよびコードからクラスターに接続する」を参照してください。
[抽出およびアップロード]	抽出エージェントにより、データはファイルとして Amazon S3 バケットに保存されました。Amazon Redshift の COPY コマンドを使用して Amazon Redshift にデータをロードできます。詳細については、Amazon Redshift のドキュメントの「Amazon S3 からデータをロードする」を参照してください。 設定した抽出タスクに対応する複数のフォルダが Amazon S3 バケットにあります。データを Amazon Redshift にロードするときに、各タスクによって作成されたマニフェストファイルの名前を指定します。マニフェストファイルは、次に示すように、Amazon S3 バケットのタスクフォルダに表示されます。

移行モード	データの場所
	
[抽出のみ]	抽出エージェントにより、データはファイルとして作業フォルダに保存されました。手動で Amazon S3 バケットにデータをコピーし、[抽出およびアップロード] の手順に従います。

での仮想パーティショニングの使用 AWS Schema Conversion Tool

通常、パーティション分割されていない大きなテーブルを管理する最適な方法は、サブタスクを作成することです。これにより、フィルタリング規則を使用してテーブルデータの仮想パーティションを作成します。では AWS SCT、移行したデータ用の仮想パーティションを作成できます。データ型に対応した 3 つのパーティションタイプがあります。

- RANGE パーティションタイプは、数値データ型、日付/時刻データ型に対応します。
- LIST パーティションタイプは、数値データ型、文字データ型、日付/時刻データ型に対応します。
- DATE AUTO SPLIT パーティションタイプは、数値、日付、および時刻データ型に対応します。

AWS SCT パーティションを作成するために指定した値を検証します。たとえば、データ型 NUMERIC の列をパーティション分割しようとして、別のデータ型の値を指定すると、 AWS SCT エラーが発生します。

また、Amazon Redshift AWS SCT へのデータ移行に使用している場合は、ネイティブパーティショニングを使用して大きなテーブルの移行を管理できます。詳細については、「ネイティブパーティショニングを使用する」を参照してください。

仮想パーティション分割の作成時の制限

仮想パーティションの作成に伴う制限は以下のとおりです。

- 仮想パーティション分割を使用できるのは、パーティション分割されていないテーブルに対してのみです。
- 仮想パーティション分割は、データ移行ビューでのみ使用できます。
- 仮想パーティション分割では、UNION ALL VIEW オプションを使用できません。

RANGE パーティションタイプ

RANGE パーティションタイプは、数値データ型と日付/時刻データ型の列値の範囲に基づいてデータをパーティション分割します。このパーティションタイプによって WHERE 句が作成されます。ユーザーはパーティションごとに値の範囲を指定します。パーティション分割した列の値のリストを指定するには、 [Values] (値) ボックスを使用します。値の情報は .csv ファイルを使用してロードできます。

RANGE パーティションタイプは、パーティション値の両端にデフォルトのパーティションを作成します。これらのデフォルトパーティションは、指定されたパーティション値より小さい、または大きいすべてのデータをキャッチします。

例えば、指定した値の範囲に基づいて複数のパーティションを作成できます。次の例では、LO_TAX のパーティション分割値を指定して複数のパーティションを作成しています。

```
Partition1: WHERE LO_TAX <= 10000.9
Partition2: WHERE LO_TAX > 10000.9 AND LO_TAX <= 15005.5
Partition3: WHERE LO_TAX > 15005.5 AND LO_TAX <= 25005.95
```

RANGE 仮想パーティションを作成するには

1. AWS SCT開きます。

2. [データ移動ビュー (その他)] モードを選択します。
3. 仮想パーティション分割を設定するテーブルを選択します。テーブルのコンテキスト (右クリック) メニューを開き、[仮想パーティションを追加] を選択します。
4. [仮想パーティションを追加] ダイアログボックスで、以下のように情報を入力します。

オプション	アクション
[パーティションタイプ]	[範囲] を選択します。選択したタイプに応じてダイアログボックスの UI が変わります。
[列名]	パーティション分割する列を選択します。
[列のタイプ]	列の値のデータ型を選択します。
[値]	新しい値を追加するために、[新しい値] ボックスに各値を入力し、プラス記号を選択して値を追加します。
[ファイルからロード]	(オプション) パーティション値を含める .csv ファイルの名前を入力します。

5. [OK] をクリックします。

LIST パーティションタイプ

LIST パーティションタイプは、数値データ型、文字データ型、日付/時刻データ型の列値に基づいてデータをパーティション分割します。このパーティションタイプによって WHERE 句が作成されます。ユーザーはパーティションごとに値を指定します。パーティション分割した列の値のリストを指定するには、 [値] ボックスを使用します。値の情報は .csv ファイルを使用してロードできます。

例えば、指定した値に基づいて複数のパーティションを作成できます。次の例では、LO_ORDERKEY のパーティション分割値を指定して複数のパーティションを作成しています。

```
Partition1: WHERE LO_ORDERKEY = 1
Partition2: WHERE LO_ORDERKEY = 2
Partition3: WHERE LO_ORDERKEY = 3
…
PartitionN: WHERE LO_ORDERKEY = USER_VALUE_N
```

指定した値の範囲に含まれない値のデフォルトパーティションを作成することもできます。

移行から特定の値を除外する場合は、LIST パーティションタイプを使用してソースデータをフィルタ処理できます。例えば、LO_ORDERKEY = 4 で行を省略するとします。この場合は、パーティション値のリストに、値 4 が含まれておらず、また、[他の値を含める] が選択されていないことを確認します。

LIST 仮想パーティションを作成するには

1. 開く AWS SCT。
2. [データ移動ビュー (その他)] モードを選択します。
3. 仮想パーティション分割を設定するテーブルを選択します。テーブルのコンテキスト (右クリック) メニューを開き、[仮想パーティションを追加] を選択します。
4. [仮想パーティションを追加] ダイアログボックスで、以下のように情報を入力します。

オプション	アクション
[パーティションタイプ]	[リスト] を選択します。選択したタイプに応じてダイアログボックスの UI が変わります。
[列名]	パーティション分割する列を選択します。
[新しい値]	パーティション分割値のセットに追加する値をここに入力します。
[他の値を含める]	パーティション分割条件に適合しないすべての値を保存するデフォルトパーティションを作成する場合は、このオプションを選択します。
[ファイルからロード]	(オプション) パーティション値を含める .csv ファイルの名前を入力します。

5. [OK] をクリックします。

DATE AUTO SPLIT パーティションタイプ

DATE AUTO SPLIT パーティションタイプは、RANGE パーティションを自動的に生成する方法です。DATA AUTO SPLIT では、パーティショニング属性、開始位置と終了位置、値間の範囲のサイズを指定します。 AWS SCT その後 AWS SCT で、パーティション値が自動的に計算されます。

DATA AUTO SPLIT は、範囲パーティションの作成に関連する多くの作業を自動化します。この手法と範囲パーティショニングのトレードオフは、パーティションの境界に対してどれだけの制御が必要になるかということです。自動分割プロセスでは、常に同じサイズ (均一) の範囲が作成されます。範囲パーティショニングでは、特定のデータ分散に必要な範囲のサイズを変化させることができます。例えば、毎日、毎週、隔週、毎月などを使用できます。

```
Partition1: WHERE LO_ORDERDATE >= ‘1954-10-10’ AND LO_ORDERDATE < ‘1954-10-24’
Partition2: WHERE LO_ORDERDATE >= ‘1954-10-24’ AND LO_ORDERDATE < ‘1954-11-06’
Partition3: WHERE LO_ORDERDATE >= ‘1954-11-06’ AND LO_ORDERDATE < ‘1954-11-20’
…
PartitionN: WHERE LO_ORDERDATE >= USER_VALUE_N AND LO_ORDERDATE <= ‘2017-08-13’
```

DATE AUTO SPLIT 仮想パーティションを作成するには

1. 開く AWS SCT。
2. [データ移動ビュー (その他)] モードを選択します。
3. 仮想パーティション分割を設定するテーブルを選択します。テーブルのコンテキスト (右クリック) メニューを開き、[仮想パーティションを追加] を選択します。
4. [仮想パーティションを追加] ダイアログボックスで、以下のように情報を入力します。

オプション	アクション
[パーティションタイプ]	[日付の自動分割] を選択します。選択したタイプに応じてダイアログボックスの UI が変わります。
[列名]	パーティション分割する列を選択します。
[開始日]	開始日を入力します。
[終了日]	終了日を入力します。
[間隔]	間隔の単位を入力し、その単位の値を選択します。

5. [OK] をクリックします。

ネイティブパーティショニングを使用する

データ移行をスピードアップするために、データ抽出エージェントはソースデータウェアハウスサーバー上のテーブルのネイティブパーティションを使用できます。 AWS SCT Greenplum、Netezza、Oracle から Amazon Redshift への移行のためのネイティブパーティショニングをサポートします。

例えば、プロジェクトを作成した後、スキーマの統計を収集し、移行対象として選択したテーブルのサイズを分析することができます。指定されたサイズを超えるテーブルでは、ネイティブパーティショニングメカニズムがトリガーされます。 AWS SCT

ネイティブパーティショニングを使用するには

1. 開き AWS SCT、[ファイル] の [新規プロジェクト] を選択します。[新しいプロジェクト] ダイアログボックスが表示されます。
2. 新しいプロジェクトを作成し、ソースとターゲットサーバーを追加し、マッピングルールを作成します。詳細については、「AWS SCT プロジェクトの作成」を参照してください。
3. [ビュー]、[メインビュー] の順に選択します。
4. [現在のプロジェクト設定] から、[データ移行] タブを選択します。[自動パーティショニングを使う] を選択します。Greenplum と Netezza のソースデータベースでは、サポートされているテーブルの最小サイズ (100 など) をメガバイト単位で入力します。 AWS SCT は、空でないネイティブパーティションごとに、個別の移行サブタスクを自動的に作成します。Oracle から Amazon Redshift AWS SCT への移行では、すべてのパーティションテーブルにサブタスクを作成します。
5. ソースデータベースのスキーマを表示する左のパネルで、スキーマを選択します。オブジェクトのコンテキスト (右クリック) メニューを開き、[統計の収集] を選択します。Oracle から Amazon Redshift へのデータ移行の場合、このステップは省略できます。
6. 移行するテーブルをすべて選択します。
7. 必要な数のエージェントを登録します。詳細については、「抽出エージェントをに登録する AWS Schema Conversion Tool」を参照してください。
8. 選択したテーブルのデータ抽出タスクを作成します。詳細については、「AWS SCT データ抽出タスクの作成、実行、監視」を参照してください。

   大きなテーブルがサブタスクに分割されていること、および各サブタスクが、ソースデータウェアハウスの 1 つのスライスにあるテーブルの一部を示すデータセットと一致していることを確認します。

9. AWS SCT データ抽出エージェントがソーステーブルからのデータの移行を完了するまで、移行プロセスを開始して監視します。

LOB を Amazon Redshift に移行する

Amazon Redshift は、ラージバイナリオブジェクト (LOB) の保存をサポートしていません。ただし、1 つ以上の LOB を Amazon Redshift に移行する必要がある場合は、 AWS SCT 移行を実行できます。この場合、 AWS SCT は Amazon S3 バケットを使用して LOB を保存し、Amazon S3 バケットの URL を Amazon Redshift に保存された移行済みデータに書き込みます。

LOB を Amazon Redshift に移行するには

1. プロジェクトを開きます。 AWS SCT
2. ソースおよびターゲットデータベースに接続します。ターゲットデータベースのメタデータを更新し、変換されたテーブルがそこに存在することを確認します。
3. [アクション] の [ローカルタスクの作成] を選択します。
4. [移動モード] で、以下のいずれかを選択します。

   - [抽出およびアップロード] - データを抽出し、Amazon S3 にアップロードします。
   - [抽出、アップロード、コピー] - データを抽出し、Amazon S3 にアップロードして、Amazon Redshift データウェアハウスにコピーします。
5. [Amazon S3 設定] を選択します。
6. [S3 バケット LOB フォルダ] に、LOB の保存先となる S3 バケット内のフォルダの名前を入力します。

   AWS サービスプロファイルを使用する場合、このフィールドは省略可能です。 AWS SCT プロファイルのデフォルト設定を使用できます。別の Amazon S3 バケットを使用するには、ここにパスを入力します。
7. [プロキシを使用] をオンにして、 Amazon S3 にデータをアップロードします。次に、データ転送プロトコルを選択し、ホスト名、ポート、ユーザー名、パスワードを入力します。
8. [エンドポイントタイプ]では、[FIPS] を選択して連邦情報処理規格 (FIPS) エンドポイントを使用します。[VPCE] を選択して、仮想プライベートクラウド (VPC) エンドポイントを使用します。次に、[VPC エンドポイント] に VPC エンドポイントのドメインネームシステム (DNS) を入力します。
9. [Amazon S3 にファイルを保持する] オプションをオンにすると、抽出したファイルを Amazon Redshift にコピーした後も Amazon S3 に保持されます。

10. 作成を選択して、タスクを作成します。

データ抽出エージェントのベストプラクティスとトラブルシューティング

以下は、抽出エージェントを使用するためのベストプラクティスの提案およびトラブルシューティングの解決策です。

問題	トラブルシューティングの推奨事項
パフォーマンスが低い	パフォーマンスを向上させるには、以下のことをお勧めします。 • 複数のエージェントをインストールします。 • データウェアハウスに近いコンピュータにエージェントをインストールします。 • 1 つのエージェントタスクですべてのテーブルを実行しないでください。
競合による遅延	データウェアハウスに同時に多数のエージェントがアクセスすることを避けます。
エージェントが一時的にダウンする	エージェントがダウンする場合、各タスクのステータスは AWS SCTに失敗として表示されます。待機していると、場合によってエージェントは復旧できます。この場合、タスクのステータスは AWS SCTで更新します。
エージェントが完全にダウンする	エージェントを実行しているコンピュータが完全にダウンし、そのエージェントがタスクを実行している場合、新しいエージェントを代用してタスクを継続できます。元のエージェントの作業フォルダが、元のエージェントと同じコンピュータにない場合にのみ、新しいエージェントを代用できます。新しいエージェントを代用するには、以下の操作を実行します。 • 新しいコンピュータにエージェントをインストールします。 • ポート番号や作業フォルダを含め元のエージェントと同じ設定を使用して新しいエージェントを設定します。

問題	トラブルシューティングの推奨事項
	• エージェントを開始します。エージェントが起動すると、タスクによって使用可能な新しいエージェントが検出され、新しいエージェントで引き続き実行されます。

AWS SCT を使用したアプリケーション SQL の変換

エンジン間でデータベーススキーマを変換するときは、古いデータベースエンジンの代わりに新しいデータベースエンジンとやり取りするように、アプリケーションの SQL コードを更新する必要があります。変換された SQL コードは表示、分析、編集、保存できます。

AWS Schema Conversion Tool (AWS SCT) を使用して、C++、C#、Java などのアプリケーションコードの SQL コードを変換できます。Oracle から PostgreSQL への変換では、AWS SCT を使用して SQL*Plus コードを PSQL に変換できます。また、Oracle から PostgreSQL への変換では、AWS SCT を使用して C#、C++、Java、および Pro*C アプリケーションに埋め込まれた SQL コードを変換できます。

トピック

アプリケーション SQL の変換の概要

アプリケーションの SQL コードを変換するには、以下に概説している手順を実行します。

- [アプリケーション変換プロジェクトを作成する] - アプリケーション変換プロジェクトはデータベーススキーマ変換プロジェクトの子になります。データベーススキーマ変換プロジェクトごとに、1 つ以上のアプリケーション変換プロジェクトをその子として作成できます。詳細については、「AWS SCT での汎用アプリケーション変換プロジェクトの作成」を参照してください。
- [SQL コードを分析して変換する] - AWS SCT でアプリケーションを分析し、SQL コードを抽出して、変換された SQL のローカルバージョンを作成して、確認と編集に使用します。このツールでは、アプリケーションのコードは変換準備ができるまでは変更されません。詳細については、「AWS SCT での SQL コードの分析と変換」を参照してください。
- [Create an application assessment report] (アプリケーション評価レポートを作成する) - アプリケーション評価レポートでは、ソースデータベーススキーマからターゲットデータベーススキーマ

へのアプリケーションの SQL コードの変換について重要な情報を得られます。詳細については、「AWS SCT での AWS SCT 評価レポートの作成と使用」を参照してください。

- [Edit, apply changes to, and save your converted SQL code] (変換された SQL コードを編集、変更適用、保存する) - 評価レポートには、自動変換できない SQL コード項目のリストが含まれます。これらの項目の変換を実行するように、SQL コードを手動で編集できます。詳細については、「AWS SCT で変換された SQL コードの編集と保存」を参照してください。

AWS SCT を使用したアプリケーション内の SQL コードの変換は、次の方法で行います。

AWS SCT を使用して、アプリケーションに埋め込まれた SQL コードを変換できます。汎用 AWS SCT アプリケーションコンバータは、アプリケーションコードをプレーンテキストとして扱います。アプリケーションコードをスキャンし、正規表現を使用して SQL コードを抽出します。このコンバータは、さまざまなタイプのソースコードファイルをサポートし、あらゆるプログラミング言語で記述されたアプリケーションコードで動作します。

汎用アプリケーションコンバータには次の制約事項があります。アプリケーションのプログラミング言語に固有のアプリケーションロジックを深く掘り下げることはありません。また、汎用コンバータは、関数、パラメータ、ローカル変数など、さまざまなアプリケーションオブジェクトの SQL ステートメントをサポートしていません。

アプリケーションの SQL コード変換を改善するには、言語固有のアプリケーション SQL コードコンバータを使用してください。詳細については、C# アプリケーションでの SQL コードの変換、Java アプリケーションの SQL コードの変換、および Pro*C アプリケーションの SQL コードの変換 を参照してください。

AWS SCT での汎用アプリケーション変換プロジェクトの作成

AWS Schema Conversion Tool では、アプリケーション変換プロジェクトはデータベーススキーマ変換プロジェクトの子になります。データベーススキーマ変換プロジェクトごとに、1 つ以上のアプリケーション変換プロジェクトをその子として作成できます。

> **Note**
>
> AWS SCT では、次のソースとターゲット間の変換はサポートされていません。
>
> - Oracle から Oracle へ

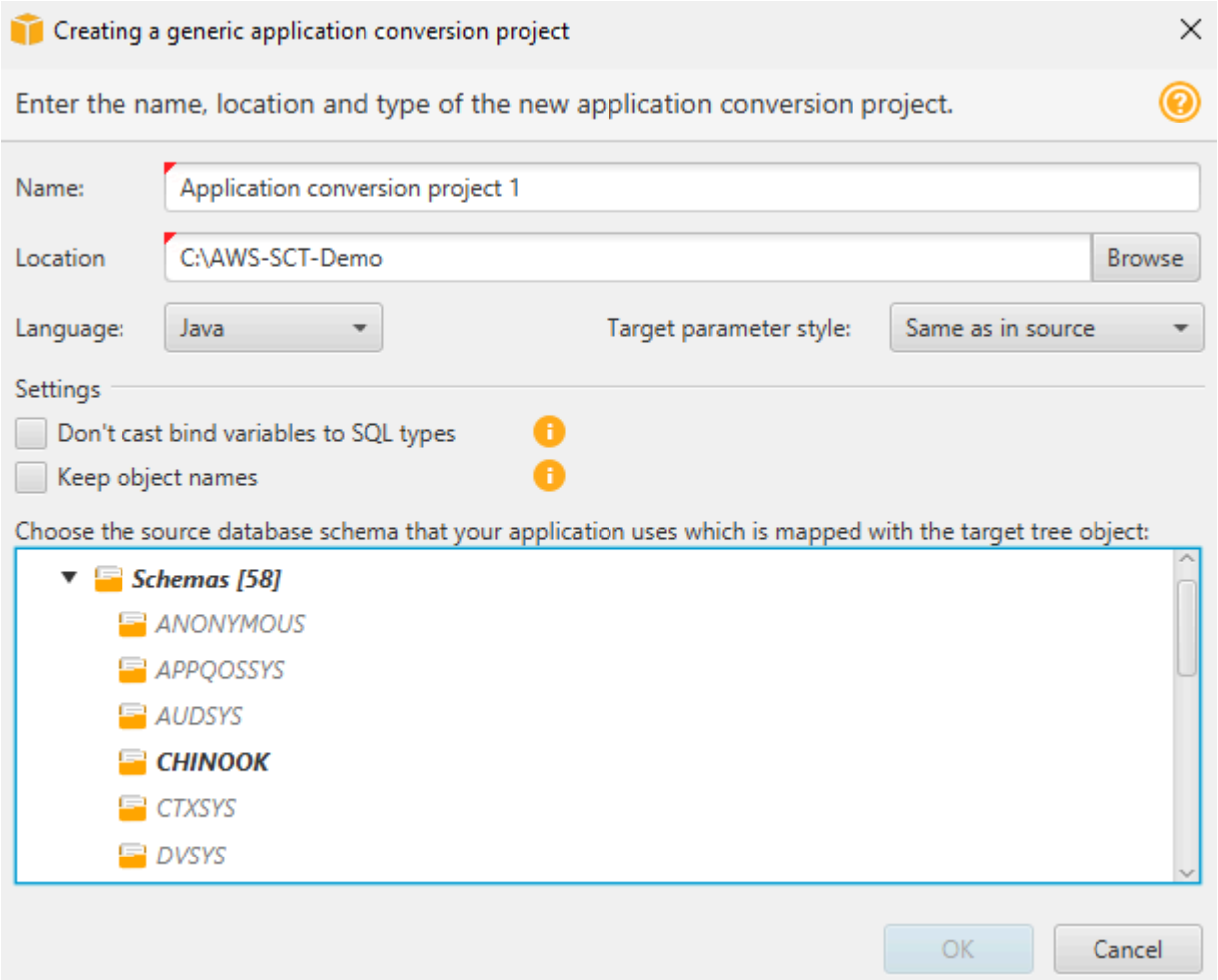
- PostgreSQL から PostgreSQL または Aurora PostgreSQL へ
- MySQL から MySQL へ
- SQL サーバーから SQL サーバーへ
- Amazon Redshift から Amazon Redshift へ
- SQL サーバーから Babelfish へ
- SQL Server Integration Services から AWS Glue へ
- Apache Cassandra から Amazon DynamoDB へ

以下の手順を使用して、汎用アプリケーション変換プロジェクトを作成します。

アプリケーション変換プロジェクトを作成するには

1. AWS Schema Conversion Tool で、[アプリケーション] メニューから [新しい汎用アプリケーション] を選択します。

   [新しいアプリケーション変換プロジェクト] ダイアログボックスが表示されます。



2. 以下のプロジェクト情報を追加します。

このパラメータについて	この操作を行います
名前	アプリケーション変換プロジェクトの名前を入力します。データベーススキーマ変換プロジェクトごとに 1 つ以上の子アプリケーション変換プロジェクトを作成できるため、後でプロジェクトを追加する場合を考慮した名前を選択します。
場所	アプリケーションのソースコードの場所を入力します。
言語	次のいずれかを選択します。 • Java • C++ • C#

このパラメータについて	この操作を行います
	• すべて
ターゲットパラメータスタイル	変換されたコードのバインド変数に使用する構文を選択します。データベースプラットフォームが異なれば、バインド変数の構文も異なります。以下のオプションのいずれかを選択します。 • Same as in source • Positional (?) • Indexed (:1) • Indexed ($1) • Named (@name) • Named (:name) • Named (&name) • Named ($name) • Named (#name) • Named (!name!)
[ソースデータベーススキーマの選択]	ソースツリーで、アプリケーションによって使用されるスキーマを選択します。このスキーマが、マッピングルールに含まれていることを確認してください。

3. バインド変数型が SQL 型に変換されないようにするには、[バインド変数型を SQL 型にキャストしない] を選択します。このオプションは、Oracle から PostgreSQL に変換する場合にのみ使用できます。

   たとえば、ソースアプリケーションコードに次の Oracle クエリが含まれているとします。

   ```
   SELECT * FROM ACCOUNT WHERE id = ?
   ```

   [バインド変数を SQL 型にキャストしない] を選択すると、AWS SCT はこのクエリを以下のように変換します。

   ```
   SELECT * FROM account WHERE id = ?
   ```

[バインド変数を SQL 型にキャストしない] をオフに すると、AWS SCT はバインド変数の型を NUMERIC データ型に変更します。変換結果は次のように表示されます。

```
SELECT * FROM account WHERE id = (?)::NUMERIC
```

4. 変換されたオブジェクトの名前にスキーマ名が追加されないようにするには、[オブジェクト名を保存] を選択します。このオプションは、Oracle から PostgreSQL に変換する場合にのみ使用できます。

   たとえば、ソースアプリケーションコードに次の Oracle クエリが含まれているとします。

   ```
   SELECT * FROM ACCOUNT
   ```

   [オブジェクト名を保存] を選択すると、AWS SCT はこのクエリを以下のように変換します。

   ```
   SELECT * FROM account
   ```

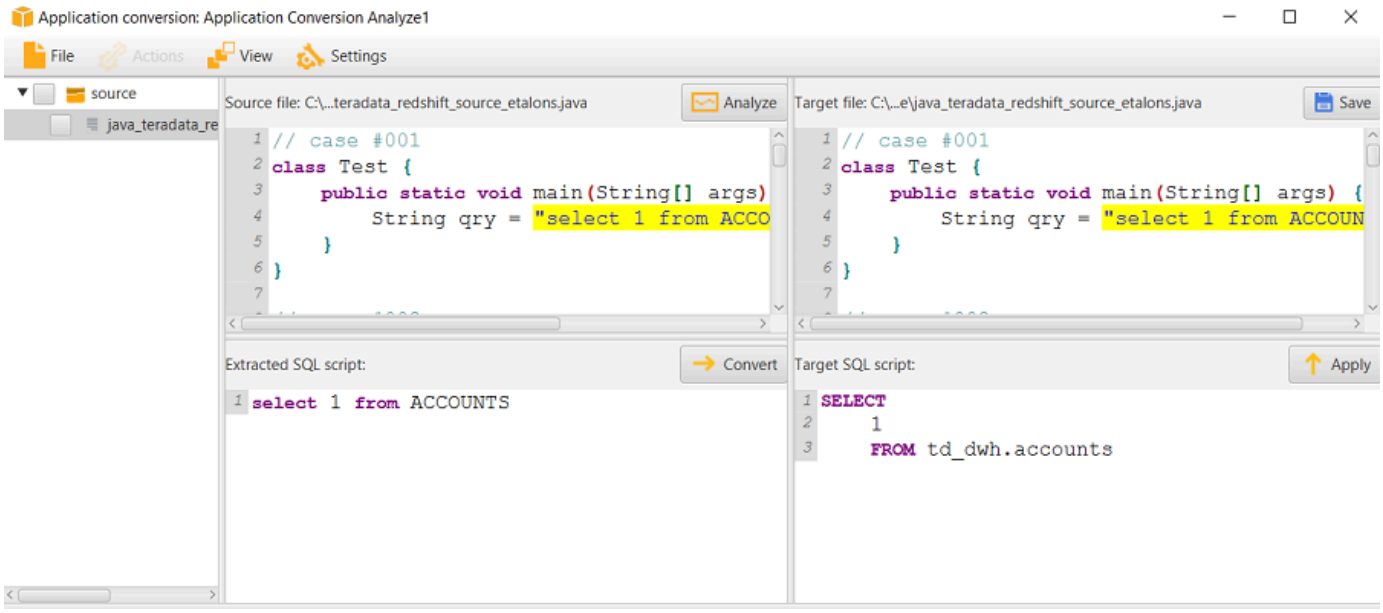
   [オブジェクト名を保存] をオフにすると、AWS SCT はテーブル名にスキーマ名を追加します。変換結果は次のように表示されます。

   ```
   SELECT * FROM schema_name.account
   ```

   ソースコードのオブジェクト名に親オブジェクトの名前が含まれている場合、AWS SCT は変換されたコードでこの形式を使用します。この場合、[オブジェクト名を保存] オプションは無視してください。AWS SCT では、変換後のコードには親オブジェクトの名前が追加されるからです。

5. [OK] を選択して、アプリケーション変換プロジェクトを作成します。

   プロジェクトウィンドウが開きます。



AWS SCT でのアプリケーション変換プロジェクトの管理

既存のアプリケーション変換プロジェクトを開いて、複数のアプリケーション変換プロジェクトを追加できます。

アプリケーション変換プロジェクトを初めて作成すると、プロジェクトウィンドウが自動的に開きます。アプリケーション変換プロジェクトウィンドウを閉じて、後で戻ることができます。

既存のアプリケーション変換プロジェクトを開くには

1. 左側のパネルでアプリケーション変換プロジェクトノードを選択し、コンテキスト (右クリック) メニューを開き、コンテキスト (右クリック) メニューを開きます。
2. [アプリケーションの作成] を選択します。

アプリケーション変換プロジェクトを追加するには

1. 左側のパネルでアプリケーション変換プロジェクトノードを選択し、コンテキスト (右クリック) メニューを開き、コンテキスト (右クリック) メニューを開きます。
2. [新しいアプリケーション] を選択します。
3. 新しいアプリケーション変換プロジェクトの作成に必要な情報を入力します。詳細については、「アプリケーション変換プロジェクトの作成」を参照してください。

AWS SCT での SQL コードの分析と変換

以下の手順で、AWS Schema Conversion Tool で SQL コードを分析して変換します。

SQL コードを変換して分析するには

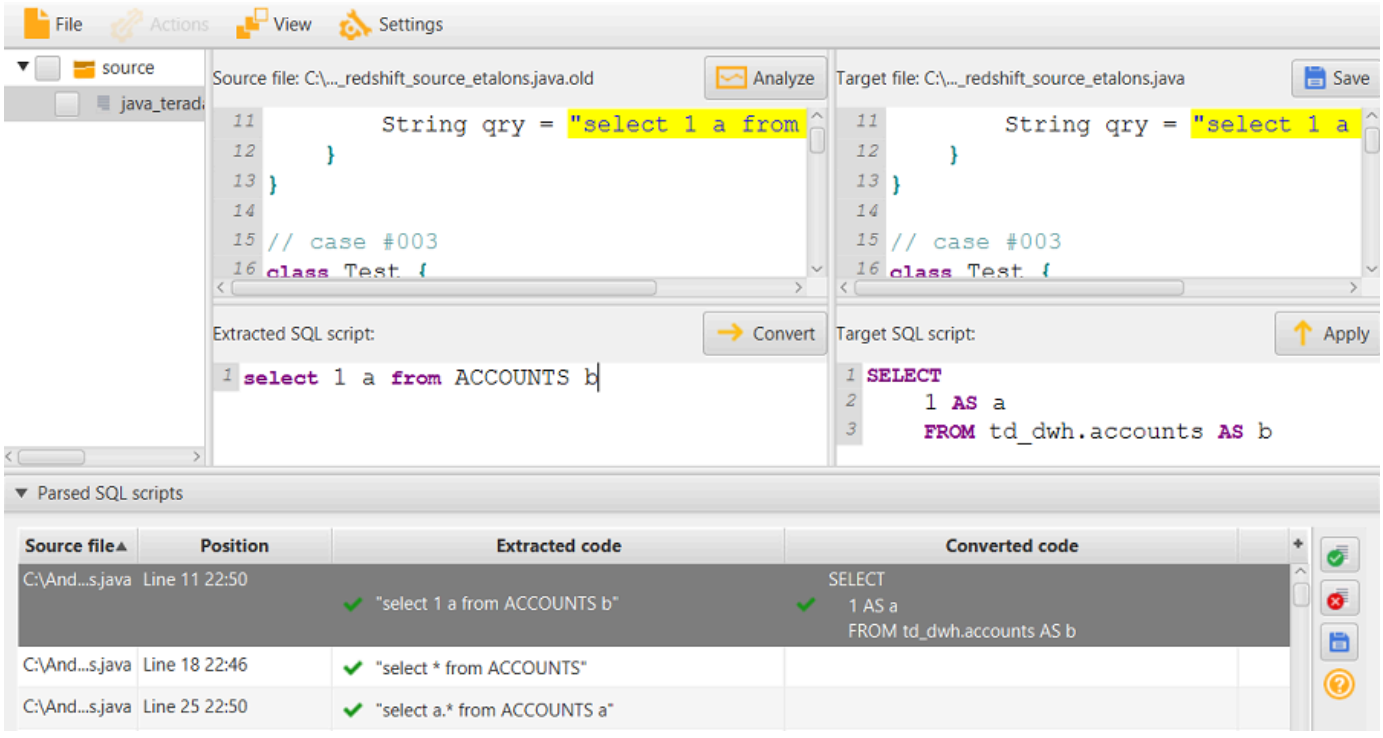
1. 既存のアプリケーション変換プロジェクトを開き、[分析] を選択します。

   AWS SCT アプリケーションコードを分析し、SQL コードを抽出します。 AWS SCT は、抽出された SQL コードを [解析済み SQL スクリプト] リストに表示します。

2. [解析済み SQL スクリプト] の場合は、抽出された SQL コードを確認する項目を選択します。AWS SCT は、選択した項目のコードを [抽出 SQL スクリプト] ペインに表示します。

3. [変換] を選択して、[抽出済み SQL スクリプト] ペインの SQL コードを変換します。 AWS SCT は、コードをターゲットデータベースと互換性のある形式に変換します。

   変換された SQL コードは編集できます。詳細については、「変換された SQL コードの編集と保存」を参照してください。



4. アプリケーション変換評価レポートを作成すると、AWS SCT は、抽出されたすべての SQL コード項目を変換します。詳細については、「評価レポートの作成と使用」を参照してください。

AWS SCT での AWS SCT 評価レポートの作成と使用

アプリケーション変換評価レポートには、アプリケーションの SQL コードをターゲットデータベースと互換性のある形式に変換する方法についての情報が記載されています。レポートには、抽出されたすべての SQL コード、変換されたすべての SQL コード、および AWS SCT が変換できない SQL コードのアクション アイテムの詳細が示されます。

アプリケーション変換評価レポートの作成

以下の手順を使用して、アプリケーション変換評価レポートを作成します。

アプリケーション変換評価レポートを作成するには

1. アプリケーション変換プロジェクトウィンドウで、[アクション] メニューから [レポートを作成] を選択します。

   AWS SCT はアプリケーション変換評価レポートを作成し、アプリケーション変換プロジェクトウィンドウで開きます。
2. [概要] タブを確認します。

   以下に示している [概要] タブには、アプリケーション評価レポートの要約が表示されます。そのタブには、自動変換されなかった SQL コード項目と自動変換された SQL コード項目が示されます。

3. [SQL 抽出アクション] を選択します。

   AWS SCT でソースコードから抽出できない SQL コード項目のリストを確認してください。
4. [SQL 変換アクション] を選択します。

   AWS SCT で自動的に変換されない SQL コード項目のリストを確認してください。推奨アクションを使用して SQL コードを手動で変換します。変換された SQL コードを編集する方法については、「AWS SCT で変換された SQL コードの編集と保存」を参照してください。
5. （オプション）レポートのローカル コピーを PDF ファイルまたはカンマ区切り値 (CSV) ファイルとして保存します。

- 右上の [PDF に保存] を選択して、レポートを PDF ファイルとして保存します。

  PDF ファイルには、エグゼクティブサマリー、アクション項目、アプリケーション変換に関する推奨事項が含まれています。

- 右上の [CSV に保存] を選択して、レポートを CSV ファイルとして保存します。

  CSV ファイルには、アクションアイテム、推奨アクション、および SQL コードの変換に必要な推定手作業の複雑さが含まれています。

AWS SCT で変換された SQL コードの編集と保存

評価レポートには、AWS SCT で自動変換されない SQL コード項目のリストが含まれます。各項目について、AWS SCT で [SQL 変換アクション] タブにアクション項目が作成されます。これらの項目の変換を実行するように、SQL コードを手動で編集できます。

以下の手順を使用して、変換された SQL コードを編集、変更適用、保存します。

変換された SQL コードを編集、変更適用、保存するには

1. [Target SQL script] (ターゲット SQL スクリプト) ペインで直接、変換された SQL コードを編集します。変換されたコードが表示されていない場合は、ペイン内をクリックすると、入力を開始できます。
2. 変換された SQL コードの編集が終了したら、[適用] を選択します。この時点で、変更はメモリに保存されますが、まだファイルには書き込まれていません。
3. [保存] を選択すると、変更がファイルに保存されます。

   [保存] を選択すると、元のファイルが上書きされます。保存前に、元のアプリケーションコードの記録用に元のファイルのコピーを作成します。

AWS SCT で、C# アプリケーションの SQL コードを次のように変換します。

Oracle から PostgreSQL への変換では、AWS Schema Conversion Tool (AWS SCT) を使用して C# アプリケーションに埋め込まれた SQL コードを変換できます。この固有の C# アプリケーションコンバータは、アプリケーションロジックを理解します。関数、パラメータ、ローカル変数など、さまざまなアプリケーションオブジェクトにあるステートメントを収集します。

このような詳細な分析により、C# アプリケーション SQL コードコンバータの方が、汎用コンバータよりも優れた変換結果が得られます。

AWS SCT での C# アプリケーション変換プロジェクトの作成

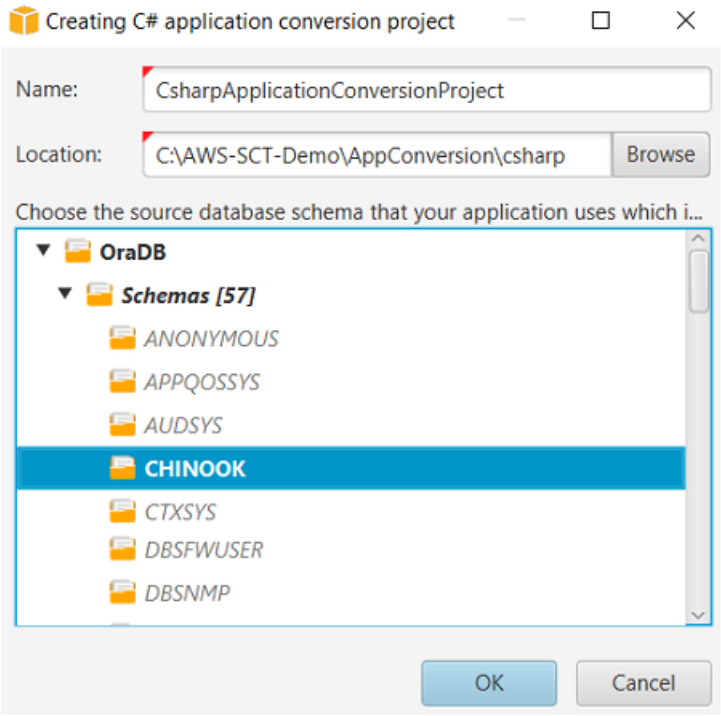
C# アプリケーション変換プロジェクトは、Oracle データベーススキーマを PostgreSQL データベーススキーマに変換する場合にのみ作成できます。必ず、ソース Oracle スキーマとターゲット PostgreSQL データベースを含むマッピングルールをプロジェクトに追加してください。詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。

1 つの AWS SCT プロジェクトに複数のアプリケーション変換プロジェクトを追加できます。以下の手順を使用して、C# アプリケーション変換プロジェクトを作成します。

C# アプリケーション変換プロジェクトを作成するには

1. データベース変換プロジェクトを作成し、ソース Oracle データベースを追加します。詳細については、「AWS SCT プロジェクトの作成」および「AWS SCT プロジェクトへのデータベースサーバーの追加」を参照してください。
2. ソース Oracle データベースとターゲット PostgreSQL データベースを含むマッピングルールを追加します。マッピングルールでターゲット PostgreSQL データベースを追加するか、仮想 PostgreSQL ターゲットデータベースプラットフォームを使用できます。詳細については、「AWS SCT でのマッピングルールの作成」および「仮想ターゲットの使用」を参照してください。
3. [ビュー] メニューで、[メインビュー] を選択します。
4. [アプリケーション] メニューから [新しい C# アプリケーション] を選択します。

   [新しいアプリケーション変換プロジェクト] ダイアログボックスが表示されます。



5. [名前] に、C# アプリケーション変換プロジェクトの名前を入力します。それぞれのデータベーススキーマ変換プロジェクトごとに 1 つ以上の子アプリケーション変換プロジェクトを作成できるため、後で複数のプロジェクトを追加する場合を考慮した名前を選択します。
6. アプリケーションのソースコードの [場所] を入力します。
7. ソースツリーで、アプリケーションによって使用されるスキーマを選択します。このスキーマが、マッピングルールに含まれていることを確認してください。AWS SCT で、マッピングルールに含まれるスキーマが強調表示されます。
8. [OK] を選択して、C# アプリケーション変換プロジェクトを作成します。
9. 左側のパネルの [アプリケーション] ノードで C# アプリケーション変換プロジェクトを探します。

AWS SCT での C# アプリケーションの SQL コードの変換

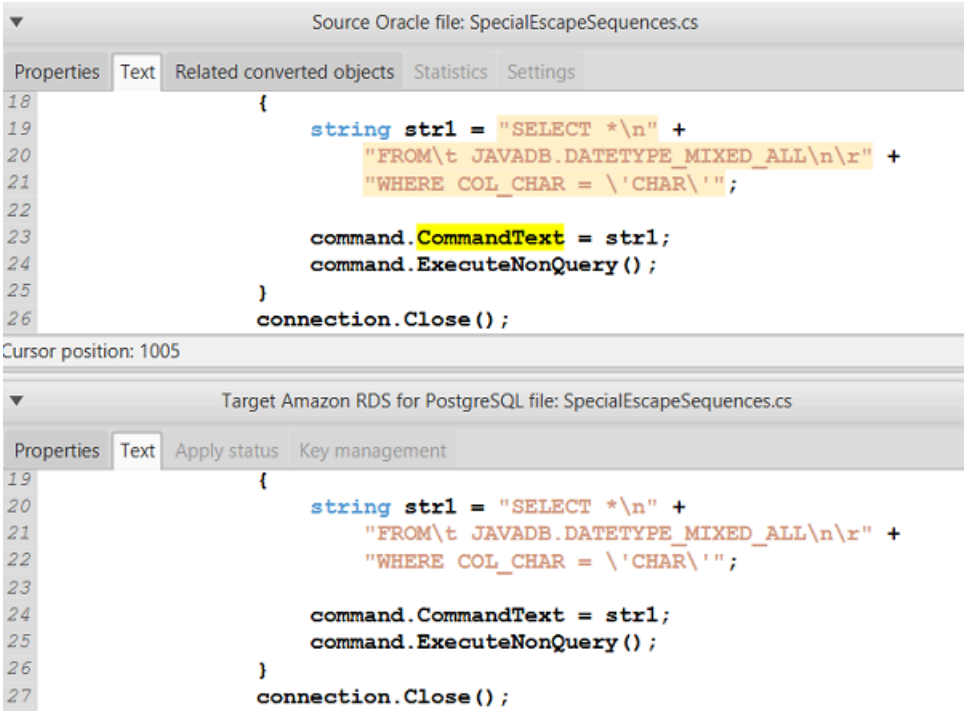
C# アプリケーションを AWS SCT プロジェクトに追加したら、このアプリケーションの SQL コードをターゲットデータベースプラットフォームと互換性のある形式に変換します。以下の手順で、AWS Schema Conversion Tool の C# アプリケーションで SQL コードを分析して変換します。

SQL コードを変換するには

1. 左側のパネルの [アプリケーション] の [C#] ノードを展開します。

2. 変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [変換] を選択します。 AWS SCT はソースコードファイルを分析し、アプリケーションロジックを決定し、コードメタデータをプロジェクトに読み込みます。このコードメタデータには、C# クラス、オブジェクト、メソッド、グローバル変数、インターフェイスなどが含まれます。

   ターゲットデータベースパネルで、AWS SCT はソースアプリケーションプロジェクトと同様のフォルダー構造を作成します。変換されたアプリケーションコードをここで確認できます。



4. 変換したアプリケーションコードを保存します。詳細については、「変換されたアプリケーションコードを保存する」を参照してください。

C# アプリケーションには、さまざまなソースデータベースとやり取りする SQL コードが含まれている場合があります。これらのソースデータベースのいくつかを PostgreSQL に移行できます。この場合、移行範囲から除外したデータベースとやりとりする SQL コードを変換しないようにしてください。C# アプリケーションのソースファイルは変換スコープから除外できます。そのためには、変換スコープから除外するファイル名のチェックボックスをオフにします。

変換スコープを変更した後も、AWS SCT は C# アプリケーションのすべてのソースファイルの SQL コードを分析します。次に、AWS SCT は変換スコープから除外したすべてのソースファイルをターゲットフォルダーにコピーします。この操作により、変換したアプリケーションファイルを保存した後でアプリケーションを構築できます。

変換したアプリケーションコードを AWS SCT で保存します。

次の手順に従って、変換されたアプリケーションコードを保存します。

変換されたアプリケーションコードを保存するには

1. ターゲットデータベースパネルの [アプリケーション] の下の [C#] ノードを展開します。
2. 変換したアプリケーションを選択し、[保存] を選択します。
3. 変換したアプリケーションコードを保存するフォルダへのパスを入力し、[フォルダを選択] を選択します。

AWS SCT での C# アプリケーション変換プロジェクトの管理

複数の C# アプリケーション変換プロジェクトを追加したり、AWS SCT プロジェクト内のアプリケーションコードを更新したり、AWS SCT プロジェクトから C# 変換プロジェクトを削除したりできます。

C# アプリケーション変換プロジェクトを追加するには

1. 左パネルの [アプリケーション] ノードを展開します。
2. [C#] ノードを選択して、コンテキスト (右クリック) メニューを開きます。
3. [新しいアプリケーション] を選択します。
4. C# アプリケーション変換プロジェクトの新規作成に必要な情報を入力します。詳細については、「C# アプリケーション変換プロジェクトの作成」を参照してください。

ソースアプリケーションコードに変更を加えたら、それを AWS SCT プロジェクトにアップロードします。

アプリケーションコードを更新するには

1. 左側のパネルの [アプリケーション] の [C#] ノードを展開します。
2. 更新するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [更新] を選択し、[はい] を選択します。

   AWS SCT は、ソースファイルからアプリケーションコードをアップロードし、変換結果を削除します。AWS SCT で行ったコード変更と変換結果を保持するには、新しい C# 変換プロジェクトを作成します。

C# アプリケーション変換プロジェクトを削除するには

1. 左側のパネルの [アプリケーション] の [C#] ノードを展開します。
2. 削除するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [削除] を選択し、[OK] を選択します。

AWS SCT での C# アプリケーション変換評価レポートの作成

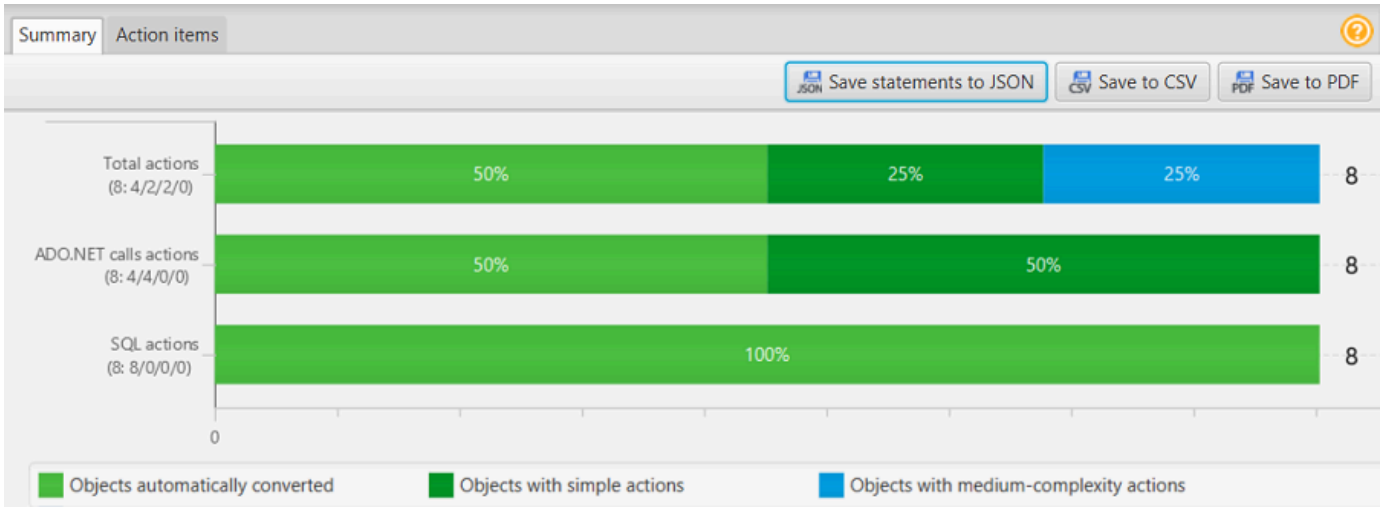
C# アプリケーション変換評価レポートには、C# アプリケーションに埋め込まれている SQL コードをターゲットデータベースと互換性のある形式に変換する方法に関する情報が記載されています。評価レポートには、すべての SQL 実行ポイントとすべてのソースコードファイルの変換詳細が記載されています。評価レポートには、AWS SCT で変換できない SQL コードのアクションアイテムも含まれています。

以下の手順を使用して、C# アプリケーション変換評価レポートを作成します。

C# アプリケーション評価レポートを作成するには

1. 左側のパネルの [アプリケーション] の [C#] ノードを展開します。
2. 変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [変換] を選択します。
4. [ビュー] メニューで [評価レポートビュー] を選択します。
5. [概要] タブを表示します。

   以下に示す [概要] タブには、C# アプリケーション評価レポートのエグゼクティブサマリーが表示されます。すべての SQL 実行ポイントとすべてのソースコードファイルの変換結果が表示されます。



6. C# アプリケーションから抽出した SQL コードを JSON ファイルとして保存するには、[ステートメントを JSON に保存] を選択します。
7. （オプション）レポートのローカル コピーを PDF ファイルまたはカンマ区切り値 (CSV) ファイルとして保存します。

   - 右上の [PDF に保存] を選択して、レポートを PDF ファイルとして保存します。

     PDF ファイルには、エグゼクティブサマリー、アクション項目、アプリケーション変換に関する推奨事項が含まれています。
   - 右上の [CSV に保存] を選択して、レポートを CSV ファイルとして保存します。

     CSV ファイルには、アクションアイテム、推奨アクション、および SQL コードの変換に必要な推定手作業の複雑さが含まれています。

AWS SCT での C++ アプリケーションの SQL コードの変換には

Oracle から PostgreSQL への変換では、AWS SCT を使用して C++ アプリケーションに埋め込まれた SQL コードを変換できます。この特定の C++ アプリケーションコンバータはアプリケーションロジックを理解します。関数、パラメータ、ローカル変数など、さまざまなアプリケーションオブジェクトにあるステートメントを収集します。

このような詳細な分析により、C++ アプリケーション SQL コードコンバータの方が汎用コンバータよりも優れた変換結果が得られます。

AWS SCT での C++ アプリケーション変換プロジェクトの作成

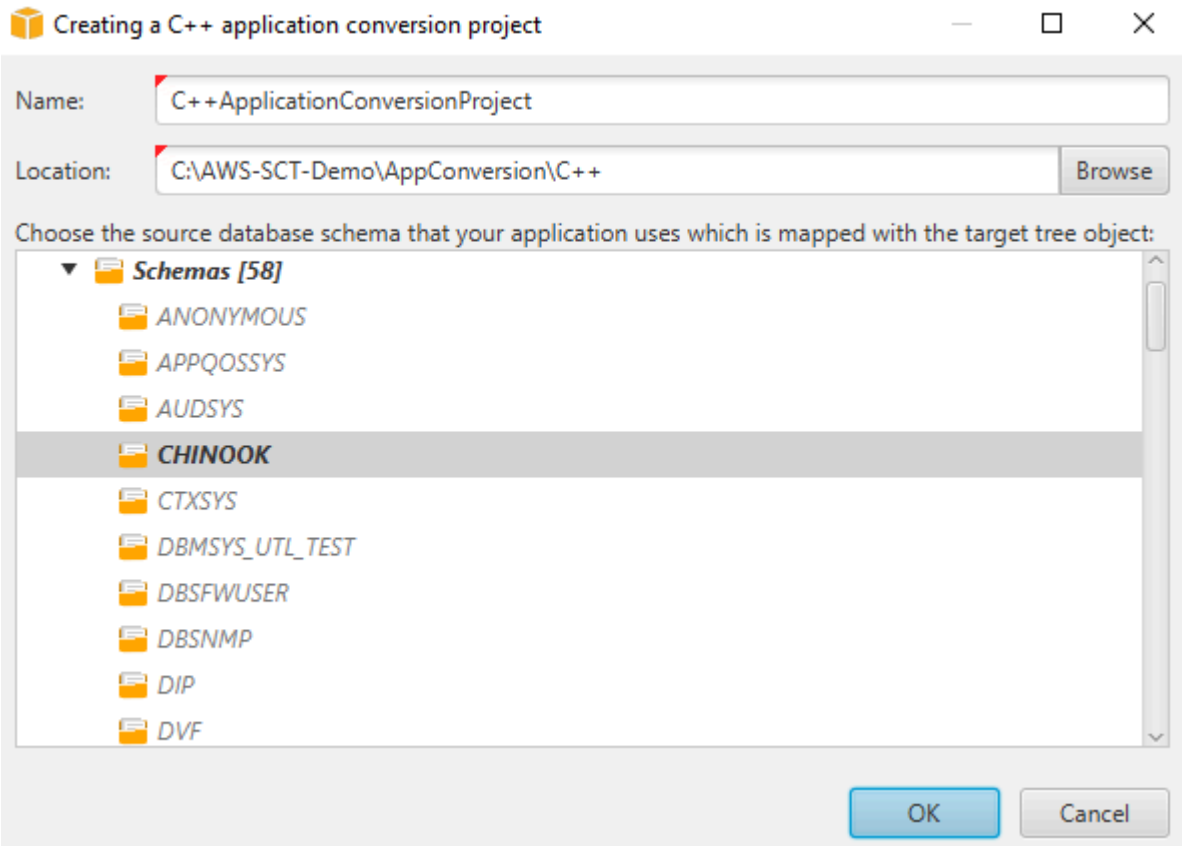
C++ アプリケーション変換プロジェクトは、Oracle データベーススキーマを PostgreSQL データベーススキーマに変換する場合にのみ作成できます。必ず、ソース Oracle スキーマとターゲット PostgreSQL データベースを含むマッピングルールをプロジェクトに追加してください。詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。

1 つの AWS SCT プロジェクトに複数のアプリケーション変換プロジェクトを追加できます。

C++ アプリケーション変換プロジェクトを作成するには

1. データベース変換プロジェクトを作成し、ソース Oracle データベースを追加します。詳細については、「AWS SCT プロジェクトの作成」および「AWS SCT プロジェクトへのデータベースサーバーの追加」を参照してください。
2. ソース Oracle データベースとターゲット PostgreSQL データベースを含むマッピングルールを追加します。マッピングルールでターゲット PostgreSQL データベースを追加するか、仮想 PostgreSQL ターゲットデータベースプラットフォームを使用できます。詳細については、「AWS SCT でのマッピングルールの作成」および「仮想ターゲットの使用」を参照してください。
3. [ビュー] メニューで、[メインビュー] を選択します。
4. [アプリケーション] メニュー で、[新しい C++ アプリケーション] を選択します。

   [C++ アプリケーション変換プロジェクトの作成] ダイアログボックスが表示されます。



5. [名前] に、C++ アプリケーション変換プロジェクトの名前を入力します。それぞれのデータベーススキーマ変換プロジェクトごとに 1 つ以上の子アプリケーション変換プロジェクトを作成できるため、後で複数のプロジェクトを追加する場合を考慮した名前を選択します。
6. アプリケーションのソースコードの [場所] を入力します。
7. ソースツリーで、アプリケーションによって使用されるスキーマを選択します。このスキーマが、マッピングルールに含まれていることを確認してください。AWS SCT で、マッピングルールに含まれるスキーマが強調表示されます。
8. [OK] を選択して、C++ アプリケーション変換プロジェクトを作成します。
9. 左パネルの [アプリケーション] ノードで C++ アプリケーション変換プロジェクトを探します。

AWS SCT で、C++ アプリケーションの SQL コードを次のように変換します。

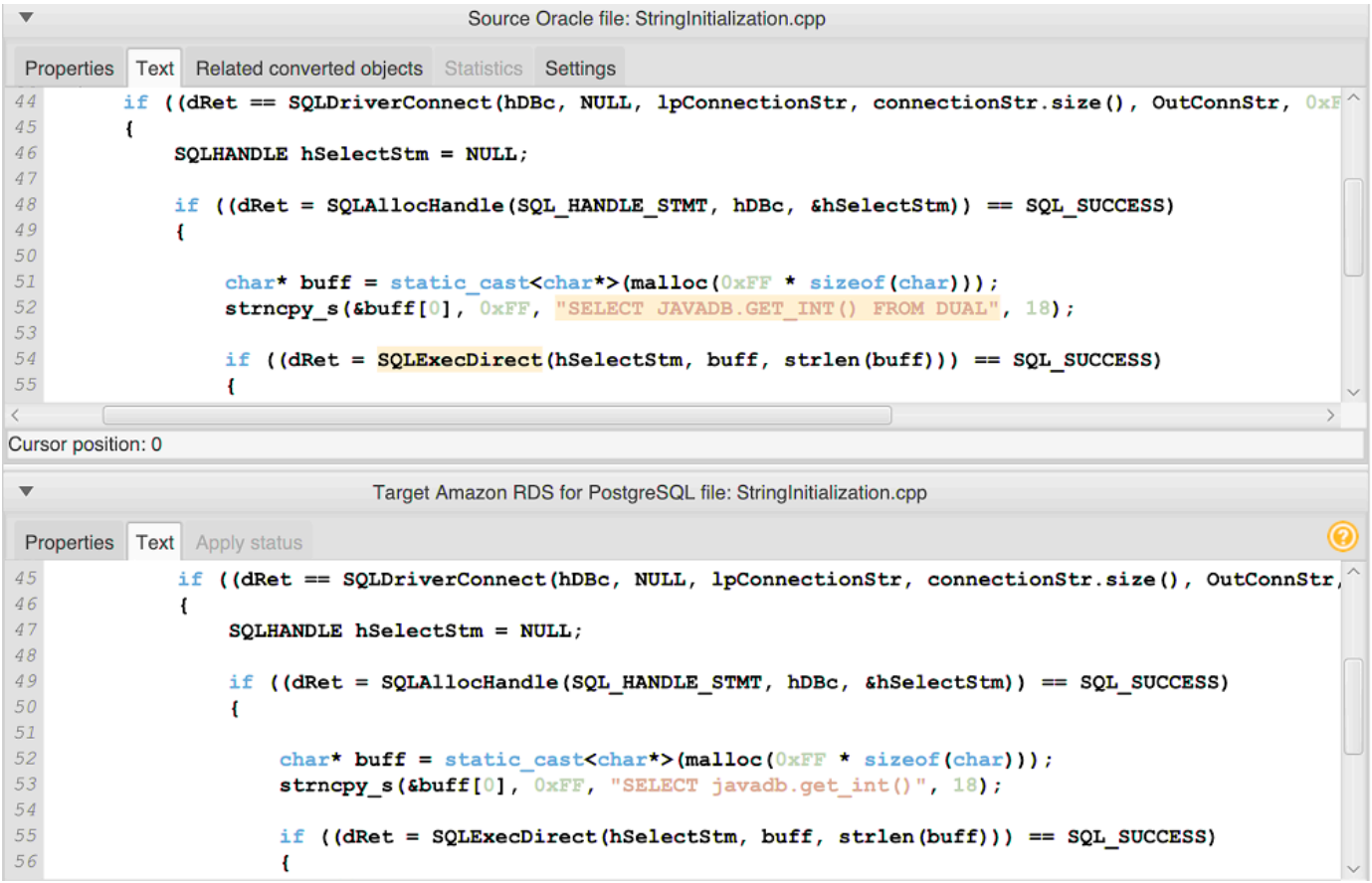
C++ AWS SCT アプリケーションをプロジェクトに追加したら、このアプリケーションの SQL コードをターゲットデータベースプラットフォームと互換性のある形式に変換します。以下の手順で、AWS SCT で C++ アプリケーションに埋め込まれている SQL コードを分析して変換します。

SQL コードを変換するには

1. 左パネルの [アプリケーション] の下の [C++] ノードを展開し、変換するアプリケーションを選択します。
2. ソース Oracle アプリケーションプロジェクトで、[設定] を選択します。選択した C++ アプリケーションの変換設定を確認して編集します。AWS SCT プロジェクトに追加したすべての C++ アプリケーションの変換設定を指定することもできます。詳細については、「C++ アプリケーション変換プロジェクトの管理」を参照してください。
3. [コンパイラタイプ] では、C++ アプリケーションのソースコードに使用するコンパイラを選択します。AWS SCT は、Microsoft Visual C++、GCC、GNU コンパイラコレクション、Clang の C++ コンパイラをサポートしています。デフォルトオプションは [Microsoft Visual C++] です。
4. ユーザー定義マクロの場合は、C++ プロジェクトのユーザー定義マクロを含むファイルへのパスを入力します。このファイルの構造が、`#define name value` であることを確認します。前の例では、`value` はオプションのパラメータです。このオプションのパラメータのデフォルト値は `1` です。

   このファイルを作成するには、Microsoft Visual Studio でプロジェクトを開き、[プロジェクト]、[プロパティ]、[C/C++]、および [プリプロセッサ] を選択します。プリプロセッサ定義の場合は、[編集] を選択し、名前と値を新しいテキストファイルにコピーします。次に、ファイル内の各文字列に、プレフィックス `#define` を追加します。
5. 外部インクルードディレクトリには、C++ プロジェクトで使用する外部ライブラリを含むフォルダへのパスを入力します。
6. 左側のペインで変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
7. [変換] を選択します。 AWS SCT はソースコードファイルを分析し、アプリケーションロジックを決定し、コードメタデータをプロジェクトに読み込みます。このコードメタデータには、C++ クラス、オブジェクト、メソッド、グローバル変数、インターフェイスなどが含まれます。

   ターゲットデータベースパネルで、AWS SCT はソースアプリケーションプロジェクトと同様のフォルダ構造を作成します。ここでは、次に示すように、変換されたアプリケーションコードを確認できます。



8. 変換したアプリケーションコードを保存します。詳細については、「変換されたアプリケーションコードを保存する」を参照してください。

変換したアプリケーションコードを AWS SCT で保存します。

次の手順に従って、変換されたアプリケーションコードを保存します。

変換されたアプリケーションコードを保存するには

1. ターゲットデータベースパネルの[アプリケーション] の下にある [C++] ノードを展開します。
2. 変換したアプリケーションを選択し、[保存] を選択します。
3. 変換したアプリケーションコードを保存するフォルダへのパスを入力し、[フォルダを選択] を選択します。

AWS SCT での C++ アプリケーション変換プロジェクトの管理

複数の C++ アプリケーション変換プロジェクトを追加したり、変換設定を編集したり、C++ アプリケーションコードを更新したり、AWS SCT プロジェクトから C++ 変換プロジェクトを削除したりできます。

C++ アプリケーション変換プロジェクトを追加するには

1. 左パネルの [アプリケーション] ノードを展開します。
2. [C++] ノードを選択して、コンテキスト (右クリック) メニューを開きます。
3. [新しいアプリケーション] を選択します。
4. 新しい C++ アプリケーション変換プロジェクトを作成するのに必要な情報を入力します。詳細については、「C++ アプリケーション変換プロジェクトの作成」を参照してください。

AWS SCT プロジェクト内のすべての C++ アプリケーション変換プロジェクトの変換設定を指定できます。

すべての C++ アプリケーションの変換設定を編集するには

1. [設定] メニューで [プロジェクト設定] を選択し、[アプリケーション変換] を選択します。
2. [コンパイラタイプ] では、C++ アプリケーションのソースコードに使用するコンパイラを選択します。 AWS SCT は、Microsoft Visual C++、GCC、GNU コンパイラコレクション、Clang の C++ コンパイラをサポートしています。デフォルトオプションは [Microsoft Visual C++] です。
3. ユーザー定義マクロの場合は、C++ プロジェクトのユーザー定義マクロを含むファイルへのパスを入力します。このファイルの構造が、`#define name value` であることを確認します。前の例では、`value` はオプションのパラメータです。このオプションのパラメータのデフォルト値は `1` です。

   このファイルを作成するには、Microsoft Visual Studio でプロジェクトを開き、[プロジェクト]、[プロパティ]、[C/C++]、および [プリプロセッサ] を選択します。プリプロセッサ定義の場合は、[編集] を選択し、名前と値を新しいテキストファイルにコピーします。次に、ファイル内の各文字列に、プレフィックス `#define` を追加します。
4. 外部インクルードディレクトリには、C++ プロジェクトで使用する外部ライブラリを含むフォルダへのパスを入力します。
5. [OK] を選択してプロジェクト設定を保存し、ウィンドウを閉じます。

または、C++ アプリケーション変換プロジェクトごとに変換設定を指定することもできます。詳細については、「C++ アプリケーションの SQL コードの変換」を参照してください。

ソースアプリケーションコードに変更を加えたら、それを AWS SCT プロジェクトにアップロードします。

アプリケーションコードを更新するには

1. 左パネルの [アプリケーション] の下の [C++] ノードを展開します。
2. 更新するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [更新] を選択し、[はい] を選択します。

   AWS SCT は、ソースファイルからアプリケーションコードをアップロードし、変換結果を削除します。AWS SCT で行ったコード変更と変換結果を保持するには、新しい C++ 変換プロジェクトを作成します。

   また、AWS SCT で選択したアプリケーションに指定したアプリケーション変換設定も削除されます。更新したアプリケーションコードをアップロードすると、AWS SCT はプロジェクト設定のデフォルト値を適用します。

C++ アプリケーション変換プロジェクトを作成するには

1. 左パネルの [アプリケーション] の下の [C++] ノードを展開します。
2. 削除するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [削除] を選択し、[OK] を選択します。

AWS SCT での C++ アプリケーション変換評価レポートの作成

C++ アプリケーション変換評価レポートには、C++ アプリケーションに埋め込まれている SQL コードをターゲットデータベースと互換性のある形式に変換する方法に関する情報が記載されています。評価レポートには、すべての SQL 実行ポイントとすべてのソースコードファイルの変換詳細が記載されています。評価レポートには、AWS SCT で変換できない SQL コードのアクションアイテムも含まれています。

C++ アプリケーション変換評価レポートを作成するには

1. 左パネルの [アプリケーション] の下の [C++] ノードを展開します。
2. 変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。

3. [変換] を選択します。
4. [ビュー] メニューで [評価レポートビュー] を選択します。
5. [概要] タブを表示します。

   以下に示す [概要] タブには、アプリケーション評価レポートの要約が表示されます。すべての SQL 実行ポイントとすべてのソースコードファイルの変換結果が表示されます。
6. Java アプリケーションから抽出した SQL コードを JSON ファイルとして保存するには、[ステートメントを JSON に保存] を選択します。
7. （オプション）レポートのローカル コピーを PDF ファイルまたはカンマ区切り値 (CSV) ファイルとして保存します。

   - 右上の [PDF に保存] を選択して、レポートを PDF ファイルとして保存します。

     PDF ファイルには、エグゼクティブサマリー、アクション項目、アプリケーション変換に関する推奨事項が含まれています。
   - 右上の [CSV に保存] を選択して、レポートを CSV ファイルとして保存します。

     CSV ファイルには、アクションアイテム、推奨アクション、および SQL コードの変換に必要な推定手作業の複雑さが含まれています。

AWS SCT を使用した Java アプリケーションの SQL コードの変換

Oracle から PostgreSQL への変換では、AWS Schema Conversion Tool を使用して Java アプリケーションに埋め込まれた SQL コードを変換できます。この特定の Java アプリケーションコンバータはアプリケーションロジックを理解します。関数、パラメータ、ローカル変数など、さまざまなアプリケーションオブジェクトにあるステートメントを収集します。

この詳細な分析により、Java アプリケーション SQL コードコンバータは、汎用コンバータと比較して優れた変換結果を提供します。

Java アプリケーションが MyBatis フレームワークを使用してデータベースとやり取りする場合、AWS SCT は MyBatis XML ファイルやアノテーションに埋め込まれた SQL ステートメントを変換できます。これらの SQL ステートメントのロジックを理解するには、AWS SCT は MyBatis 設定ファイルを使用します。 AWS SCT は、アプリケーションフォルダ内のこのファイルを自動的に検出できます。また、このファイルへのパスを手動で入力することもできます。

AWS SCT での Java アプリケーション変換プロジェクトの作成

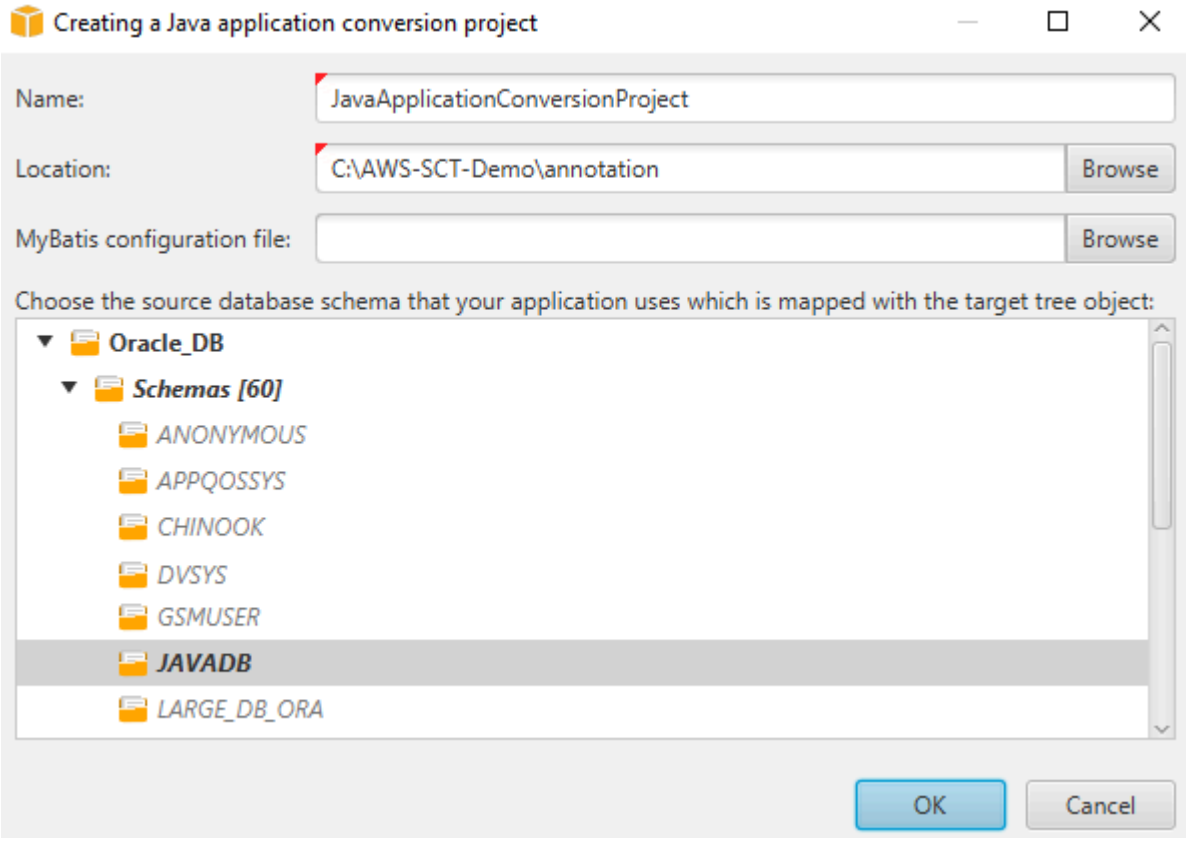
Java アプリケーション変換プロジェクトを作成できるのは、Oracle データベーススキーマを PostgreSQL データベーススキーマに変換する場合に限られます。必ず、ソース Oracle スキーマとターゲット PostgreSQL データベースを含むマッピングルールをプロジェクトに追加してください。詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。

1 つの AWS SCT プロジェクトに複数のアプリケーション変換プロジェクトを追加できます。以下の手順を使用して、Java アプリケーション変換プロジェクトを作成します。

Java アプリケーション変換プロジェクトを作成するには

1. データベース変換プロジェクトを作成し、ソース Oracle データベースを追加します。詳細については、「AWS SCT プロジェクトの作成」および「AWS SCT プロジェクトへのデータベースサーバーの追加」を参照してください。
2. ソース Oracle データベースとターゲット PostgreSQL データベースを含むマッピングルールを追加します。マッピングルールでターゲット PostgreSQL データベースを追加するか、仮想 PostgreSQL ターゲットデータベースプラットフォームを使用できます。詳細については、「AWS SCT でのマッピングルールの作成」および「仮想ターゲットの使用」を参照してください。
3. [ビュー] メニューで、[メインビュー] を選択します。
4. [アプリケーション] メニューで [新しい Java アプリケーション] を選択します。

   [新しいアプリケーション変換プロジェクト] ダイアログボックスが表示されます。



5. [名前] に、Java アプリケーション変換プロジェクトの名前を入力します。それぞれのデータベーススキーマ変換プロジェクトごとに 1 つ以上の子アプリケーション変換プロジェクトを作成できるため、後で複数のプロジェクトを追加する場合を考慮した名前を選択します。
6. アプリケーションのソースコードの [場所] を入力します。
7. (オプション) MyBatis 設定ファイルには、MyBatis 設定ファイルへのパスを入力します。 AWS SCT はアプリケーションフォルダをスキャンして、このファイルを自動的に検出します。このファイルがアプリケーションフォルダにない場合や、複数の設定ファイルを使用している場合は、パスを手動で入力します。
8. ソースツリーで、アプリケーションによって使用されるスキーマを選択します。このスキーマが、マッピングルールに含まれていることを確認してください。AWS SCT で、マッピングルールに含まれるスキーマが強調表示されます。
9. [OK] を選択して、Java アプリケーション変換プロジェクトを作成します。
10. 左側のパネルの [アプリケーション] ノードで Java アプリケーション変換プロジェクトを探します。

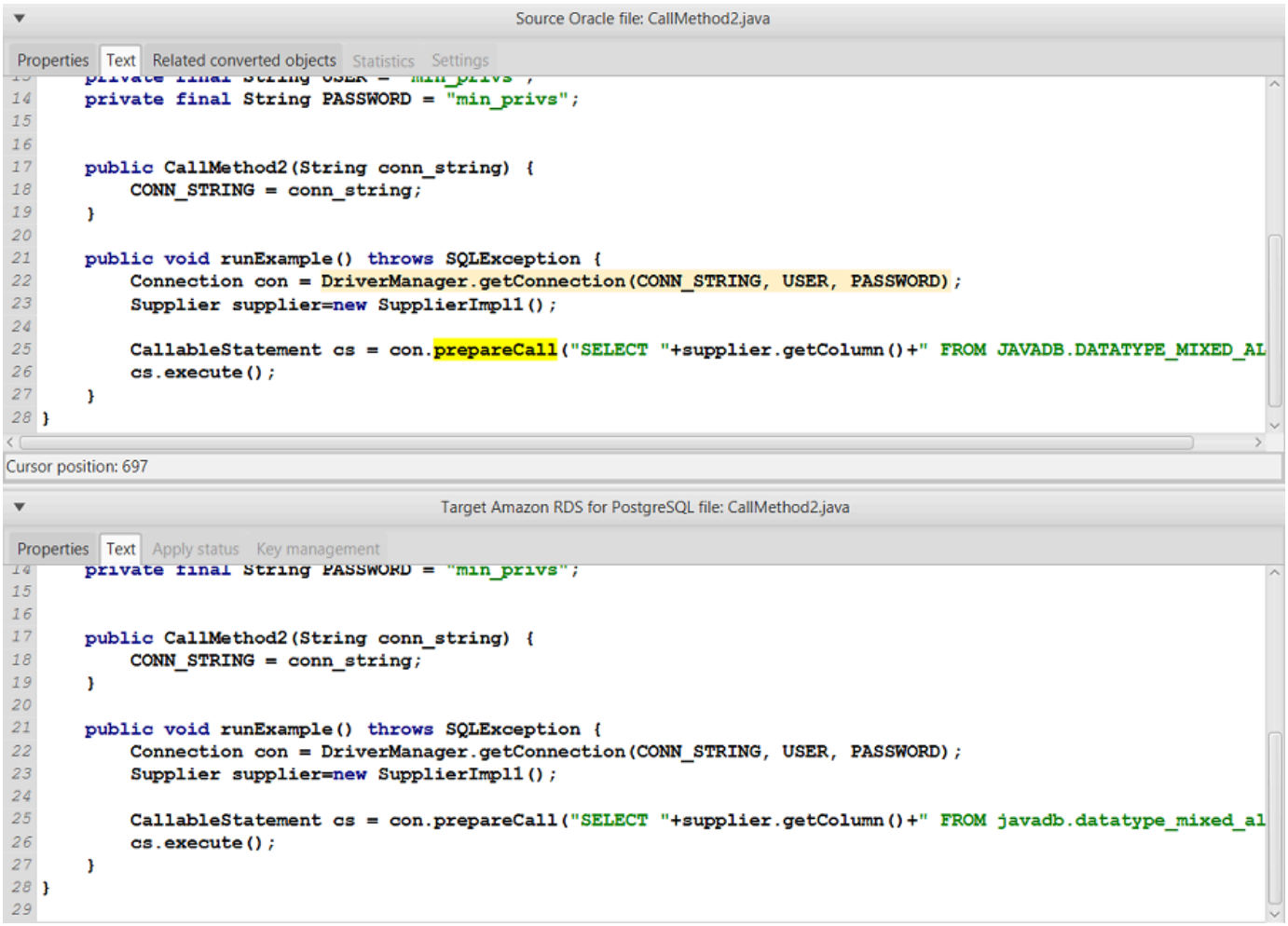
AWS SCT での Java アプリケーションの SQL コードの変換

Java アプリケーションを AWS SCT プロジェクトに追加したら、このアプリケーションの SQL コードをターゲットデータベースプラットフォームと互換性のある形式に変換します。以下の手順で、AWS Schema Conversion Tool の Java アプリケーションで SQL コードを分析して変換します。

SQL コードを変換するには

1. 左側のパネルの [アプリケーション] の下にある [Java] ノードを展開します。
2. 変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [変換] を選択します。 AWS SCT はソースコードファイルを分析し、アプリケーションロジックを決定し、コードメタデータをプロジェクトに読み込みます。このコードメタデータには、Java クラス、オブジェクト、メソッド、グローバル変数、インターフェイスなどが含まれます。

   ターゲットデータベースパネルで、AWS SCT はソースアプリケーションプロジェクトと同様のフォルダ構造を作成します。変換されたアプリケーションコードをここで確認できます。



4. 変換したアプリケーションコードを保存します。詳細については、「変換されたアプリケーションコードを保存する」を参照してください。

Java アプリケーションには、さまざまなソースデータベースとやり取りする SQL コードが含まれている場合があります。これらのソースデータベースのいくつかを PostgreSQL に移行できます。この場合、移行範囲から除外したデータベースとやりとりする SQL コードを変換しないようにしてください。Java アプリケーションのソースファイルは変換スコープから除外できます。そのためには、変換スコープから除外するファイル名のチェックボックスをオフにします。

変換スコープを変更した後も、AWS SCT はJava アプリケーションのすべてのソースファイルの SQL コードを分析します。次に、AWS SCT は変換スコープから除外したすべてのソースファイルをターゲットフォルダーにコピーします。この操作により、変換したアプリケーションファイルを保存した後でアプリケーションを構築できます。

変換したアプリケーションコードを AWS SCT で保存します。

次の手順に従って、変換されたアプリケーションコードを保存します。

変換されたアプリケーションコードを保存するには

1. ターゲットデータベースパネルの [アプリケーション] の下の [Java] ノードを展開します。
2. 変換したアプリケーションを選択し、[保存] を選択します。
3. 変換したアプリケーションコードを保存するフォルダへのパスを入力し、[フォルダを選択] を選択します。

ソース Java アプリケーションが MyBatis フレームワークを使用している場合は、新しいデータベースで動作するように、設定ファイルを更新してください。

AWS SCT での Java アプリケーション変換プロジェクトの管理

複数の Java アプリケーション変換プロジェクトを追加したり、AWS SCT プロジェクト内のアプリケーションコードを更新したり、AWS SCT プロジェクトから Java 変換プロジェクトを削除したりできます。

Java アプリケーション変換プロジェクトを追加するには

1. 左パネルの [アプリケーション] ノードを展開します。
2. [Java] ノードを選択して、のコンテキスト (右クリック) メニューを開きます。
3. [新しいアプリケーション] を選択します。
4. 新しい Java アプリケーション変換プロジェクトを作成するのに必要な情報を入力します。詳細については、「Java アプリケーション変換プロジェクトの作成」を参照してください。

ソースアプリケーションコードに変更を加えたら、それを AWS SCT プロジェクトにアップロードします。

アプリケーションコードを更新するには

1. 左側のパネルの [アプリケーション] の下にある [Java] ノードを展開します。
2. 更新するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [更新] を選択し、[はい] を選択します。

AWS SCT は、ソースファイルからアプリケーションコードをアップロードし、変換結果を削除します。AWS SCT で行ったコード変更と変換結果を保持するには、新しい Java 変換プロジェクトを作成します。

ソース Java アプリケーションが MyBatis フレームワークを使用している場合は、AWS SCT は MyBatis 設定ファイルを使用して SQL コードを解析します。このファイルを変更したら、AWS SCT プロジェクトにアップロードします。

MyBatis 設定ファイルへのパスを編集するには

1. 左側のパネルの [アプリケーション] の下にある [Java] ノードを展開します。
2. アプリケーションを選択後、[設定] を選択します。
3. [参照] を選択し、MyBatis 設定ファイルを選択します。
4. [適用] を選択します。
5. 左側のパネルでアプリケーションを選択し、コンテキスト (右クリック) メニューを開き、[更新] を選択します。

Java アプリケーション変換プロジェクトを削除するには

1. 左側のパネルの [アプリケーション] の下にある [Java] ノードを展開します。
2. 削除するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [削除] を選択し、[OK] を選択します。

AWS SCT での Java アプリケーション変換評価レポートの作成

Java アプリケーション変換評価レポートには、Java アプリケーションに埋め込まれている SQL コードをターゲットデータベースと互換性のある形式に変換する方法に関する情報が記載されています。評価レポートには、すべての SQL 実行ポイントとすべてのソースコードファイルの変換詳細が記載されています。評価レポートには、AWS SCT で変換できない SQL コードのアクションアイテムも含まれています。

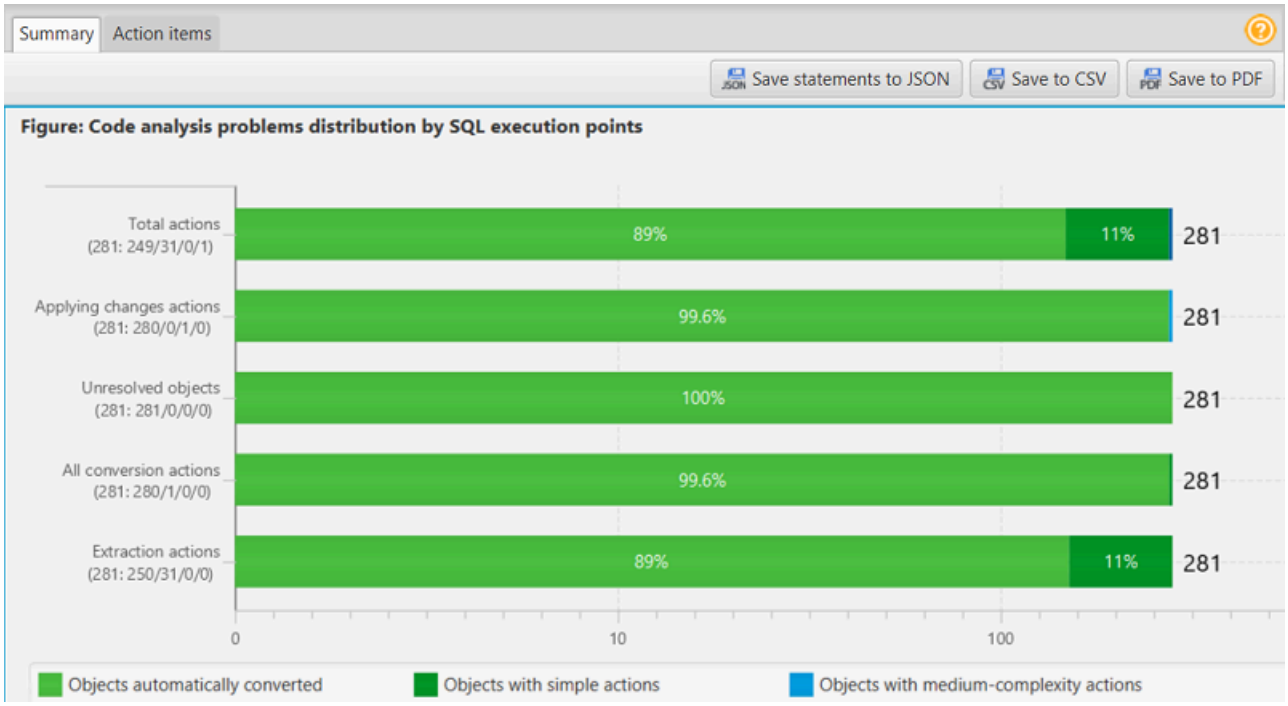
以下の手順を使用して、Java アプリケーション評価レポートを作成します。

Java アプリケーション評価レポートを作成するには

1. 左側のパネルの [アプリケーション] の下にある [Java] ノードを展開します。

2. 変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [変換] を選択します。
4. [ビュー] メニューで [評価レポートビュー] を選択します。
5. [概要] タブを確認します。

   以下に示している [概要] タブには、Java アプリケーション評価レポートの要約が表示されます。すべての SQL 実行ポイントとすべてのソースコードファイルの変換結果が表示されます。



6. Java アプリケーションから抽出した SQL コードを JSON ファイルとして保存するには、[ステートメントを JSON に保存] を選択します。
7. （オプション）レポートのローカル コピーを PDF ファイルまたはカンマ区切り値 (CSV) ファイルとして保存します。

   - 右上の [PDF に保存] を選択して、レポートを PDF ファイルとして保存します。

     PDF ファイルには、エグゼクティブサマリー、アクション項目、アプリケーション変換に関する推奨事項が含まれています。

   - 右上の [CSV に保存] を選択して、レポートを CSV ファイルとして保存します。

     CSV ファイルには、アクションアイテム、推奨アクション、および SQL コードの変換に必要な推定手作業の複雑さが含まれています。

AWS SCT を使用した Pro*C アプリケーションの SQL コードの変換

Oracle から PostgreSQL への変換では、AWS Schema Conversion Tool (AWS SCT) を使用して Pro*C アプリケーションに埋め込まれた SQL コードを変換できます。この特定の Pro*C アプリケーションコンバータはアプリケーションロジックを理解します。関数、パラメータ、ローカル変数など、さまざまなアプリケーションオブジェクトにあるステートメントを収集します。

この詳細な分析により、Pro*C アプリケーション SQL コード コンバータは、汎用コンバータと比較して優れた変換結果を提供します。

AWS SCT での Pro*C アプリケーション変換プロジェクトの作成

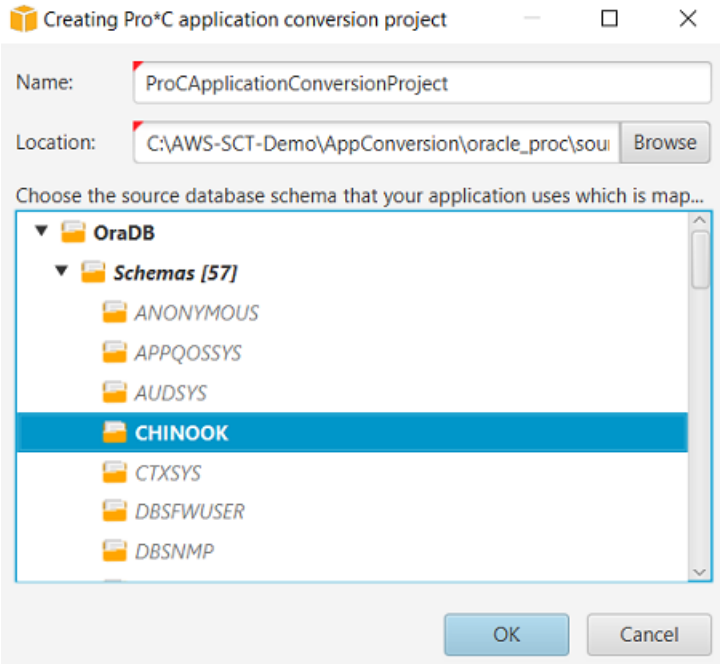
Pro*C アプリケーション変換プロジェクトは、Oracle データベーススキーマを PostgreSQL データベーススキーマに変換する場合にのみ作成できます。必ず、ソース Oracle スキーマとターゲット PostgreSQL データベースを含むマッピングルールをプロジェクトに追加してください。詳細については、「AWS SCT でのマッピングルールの作成」を参照してください。

1 つの AWS SCT プロジェクトに複数のアプリケーション変換プロジェクトを追加できます。以下の手順を使用して、Pro*C アプリケーション変換プロジェクトを作成します。

Pro*C アプリケーション変換プロジェクトを作成するには

1. データベース変換プロジェクトを作成し、ソース Oracle データベースを追加します。詳細については、「AWS SCT プロジェクトの作成」および「AWS SCT プロジェクトへのデータベースサーバーの追加」を参照してください。
2. ソース Oracle データベースとターゲット PostgreSQL データベースを含むマッピングルールを追加します。マッピングルールでターゲット PostgreSQL データベースを追加するか、仮想 PostgreSQL ターゲットデータベースプラットフォームを使用できます。詳細については、「AWS SCT でのマッピングルールの作成」および「仮想ターゲットの使用」を参照してください。
3. [ビュー] メニューで、[メインビュー] を選択します。
4. [アプリケーション] メニューで [新しい Pro*C アプリケーション] を選択します。

   [Pro*C アプリケーション変換プロジェクトの作成] ダイアログボックスが表示されます。



5. [名前] に、Pro*C アプリケーション変換プロジェクトの名前を入力します。それぞれのデータベーススキーマ変換プロジェクトごとに 1 つ以上の子アプリケーション変換プロジェクトを作成できるため、後で複数のプロジェクトを追加する場合を考慮した名前を選択します。
6. アプリケーションのソースコードの [場所] を入力します。
7. ソースツリーで、アプリケーションによって使用されるスキーマを選択します。このスキーマが、マッピングルールに含まれていることを確認してください。AWS SCT で、マッピングルールに含まれるスキーマが強調表示されます。
8. [OK] を選択して、Pro*C アプリケーション変換プロジェクトを作成します。
9. 左パネルの [アプリケーション] ノードで Pro*C アプリケーション変換プロジェクトを探します。

AWS SCT での Pro*C アプリケーションの SQL コードの変換

Pro*C AWS SCT アプリケーションをプロジェクトに追加したら、このアプリケーションの SQL コードをターゲットデータベースプラットフォームと互換性のある形式に変換します。以下の手順に従って、AWS Schema Conversion Tool で Pro*C アプリケーションに埋め込まれている SQL コードをで分析して変換します。

SQL コードを変換するには

1. 左パネルの [アプリケーション] の [Pro*C] ノードを展開します。

2. 変換するアプリケーションを選択し、[設定] を選択します。

   a. [グローバルヘッダーファイルパス] には、アプリケーションプロジェクトが使用するヘッダーファイルへのパスを入力します。

   b. [未解決のホスト変数をすべて解釈する] を選択すると、変換されたコード内の未解決変数がすべて表示されます。

   c. 変換された SQL コードで拡張パック関数を使用するには、拡張パックの [固定幅文字列変換関数を使用] を選択します。AWS SCT は、拡張パックファイルをアプリケーションプロジェクトに含めます。

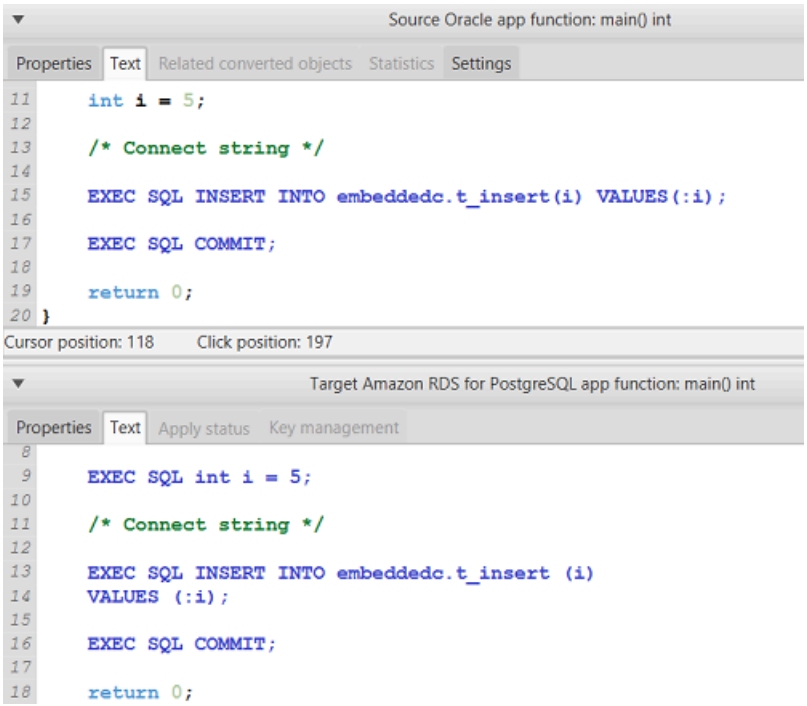
   d. [匿名 PL/SQL ブロックをスタンドアロン SQL 呼び出しまたはストアド関数に変換] を選択すると、ターゲットデータベースにすべての匿名 PL/SQL ブロック用のストアドプロシージャが作成されます。 次に、AWS SCT はこれらのストアドプロシージャの実行を変換後のアプリケーションコードに含めます。

   e. Oracle データベースカーソルの変換を改善するには、[カスタムカーソルフローを使用] を選択します。

3. 左側のパネルで変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。

4. [変換] を選択します。 AWS SCT はソースコードファイルを分析し、アプリケーションロジックを決定し、コードメタデータをプロジェクトに読み込みます。このコードメタデータには、Pro*C クラス、オブジェクト、メソッド、グローバル変数、インターフェイスなどが含まれます。

   ターゲットデータベースパネルで、AWS SCT はソースアプリケーションプロジェクトと同様のフォルダ構造を作成します。変換されたアプリケーションコードをここで確認できます。



5. 変換したアプリケーションコードを保存します。詳細については、「変換されたアプリケーションコードの編集と保存」を参照してください。

AWS SCT で変換されたアプリケーションコードの編集と保存

変換された SQL ステートメントを編集し、AWS SCT を使用してこの編集したコードを、変換された Pro*C アプリケーションコードに埋め込むことができます。次の手順に従って、変換された SQL コードを編集します。

変換された SQL コードを編集するには

1. 左パネルの [アプリケーション] の [Pro*C] ノードを展開します。
2. 変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開いてから、[変換] を選択します。
3. [ビュー] メニューで [評価レポートビュー] を選択します。
4. Pro*C アプリケーションから抽出した SQL コードを CSV ファイルとして保存するには、[ステートメントを CSV に保存] を選択します。
5. 抽出した SQL コードを保存する CSV ファイルの名前を入力し、[保存] を選択します。
6. 抽出した SQL コードを編集します。
7. [ビュー] メニューで、[メインビュー] を選択します。

8. ターゲットデータベースパネルの [アプリケーション] の下にある [Pro*C] ノードを展開します。
9. 変換したアプリケーションを選択し、コンテキスト (右クリック) メニューを開き、[CSV からステートメントをインポート] を選択します。
10. [はい] を選択し、編集した SQL コードを含むファイルを選択して [開く] を選択します。

AWS SCT は、変換された SQL ステートメントを複数の部分に分割し、ソースアプリケーションコードの適切なオブジェクトに配置します。次の手順に従って、変換されたアプリケーションコードを保存します。

変換されたアプリケーションコードを保存するには

1. ターゲットデータベースパネルの [アプリケーション] の下にある [Pro*C] ノードを展開します。
2. 変換したアプリケーションを選択し、[保存] を選択します。
3. 変換したアプリケーションコードを保存するフォルダへのパスを入力し、[フォルダを選択] を選択します。

AWS SCT での Pro*C アプリケーション変換プロジェクトの管理

複数の Pro*C アプリケーション変換プロジェクトを追加したり、AWS SCT プロジェクト内のアプリケーションコードを更新したり、Pro*C 変換プロジェクトを AWS SCT プロジェクトから削除したりできます。

Pro*C アプリケーション変換プロジェクトを追加するには

1. 左パネルの [アプリケーション] ノードを展開します。
2. コンテキスト (右クリック) メニューを開き、[Pro*C] ノードを選択します。
3. [新しいアプリケーション] を選択します。
4. 新しい Pro*C アプリケーション変換プロジェクトを作成するのに必要な情報を入力します。詳細については、「Pro*C アプリケーション変換プロジェクトの作成」を参照してください。

ソースアプリケーションコードに変更を加えたら、それを AWS SCT プロジェクトにアップロードします。

アプリケーションコードを更新するには

1. 左パネルの [アプリケーション] の [Pro*C] ノードを展開します。

2. 更新するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [更新] を選択し、[はい] を選択します。

   AWS SCT は、ソースファイルからアプリケーションコードをアップロードし、変換結果を削除します。AWS SCT で行ったコード変更と変換結果を保持するには、新しい Pro*C 変換プロジェクトを作成します。

Pro*C アプリケーション変換プロジェクトを削除するには

1. 左パネルの [アプリケーション] の [Pro*C] ノードを展開します。
2. 削除するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [削除] を選択し、[OK] を選択します。

AWS SCT で Pro*C アプリケーション変換評価レポートを作成します。

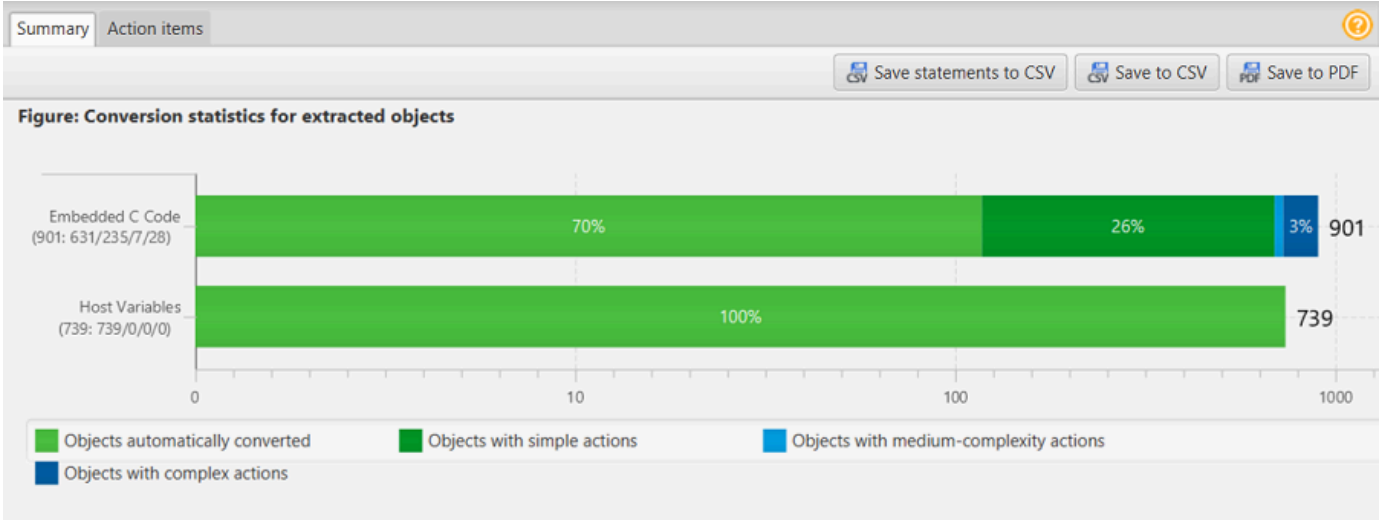
Pro*C アプリケーション変換評価レポートには、Pro*C アプリケーションに埋め込まれている SQL コードをターゲットデータベースと互換性のある形式に変換する方法に関する情報が記載されています。評価レポートには、すべての SQL 実行ポイントとすべてのソースコードファイルの変換詳細が記載されています。評価レポートには、AWS SCT で変換できない SQL コードのアクションアイテムも含まれています。

以下の手順を使用して、Pro*C アプリケーション変換評価レポートを作成します。

Pro*C アプリケーション変換評価レポートを作成するには:

1. 左パネルの [アプリケーション] の [Pro*C] ノードを展開します。
2. 変換するアプリケーションを選択し、コンテキスト (右クリック) メニューを開きます。
3. [変換] を選択します。
4. [ビュー] メニューで [評価レポートビュー] を選択します。
5. [概要] タブを確認します。

   以下に示す [概要] タブには、Pro*C アプリケーション評価レポートの要約が表示されます。すべての SQL 実行ポイントとすべてのソースコードファイルの変換結果が表示されます。



6. Pro*C アプリケーションから抽出した SQL コードを、カンマ区切り値 (CSV) ファイルとして保存するには、[ステートメントを CSV に保存] を選択します。
7. （オプション）レポートのローカル コピーを PDF ファイルまたはカンマ区切り値 (CSV) ファイルとして保存します。

   - 右上の [PDF に保存] を選択して、レポートを PDF ファイルとして保存します。

     PDF ファイルには、エグゼクティブサマリー、アクション項目、アプリケーション変換に関する推奨事項が含まれています。
   - 右上の [CSV に保存] を選択して、レポートを CSV ファイルとして保存します。

     CSV ファイルには、アクションアイテム、推奨アクション、および SQL コードの変換に必要な推定手作業の複雑さが含まれています。

AWS SCT 拡張パックの使用

AWS SCT 拡張パックは、ソースデータベースにある関数をエミュレートするアドオンモジュールです。これらの関数は、オブジェクトをターゲットデータベースに変換する際に必要です。 AWS SCT 拡張パックをインストールする前に、データベーススキーマを変換します。

AWS SCT 各拡張パックには以下のコンポーネントが含まれています。

- DB スキーマ-シーケンスなどの特定のオンライントランザクション処理 (OLTP) やオンライン分析処理 (OLAP) データベースオブジェクトをエミュレートする SQL 関数、プロシージャ、プロシージャ、プロシージャ、プロシージャ、プロシージャ、テーブルが含まれています。また、 built-in-functions ソースデータベースではサポートされていないエミュレートも行います。このスキーマの名前の形式は `aws_`*`database_engine_name`*`_ext` のようになりま。
- AWS Lambda 関数 (特定の OLTP データベース用) — ジョブのスケジューリングや電子メールの送信など、 AWS Lambda 複雑なデータベース機能をエミュレートする関数が含まれます。
- OLAP データベース用のカスタムライブラリ — Microsoft SQL Server 統合サービス (SSIS) の抽出、変換、読み込み (ETL) スクリプトをまたはに移行するために使用できる一連の Java および Python ライブラリが含まれています。 AWS Glue AWS Glue Studio

  Java ライブラリには、以下のモジュールが含まれています。

  - `spark-excel_2.11-0.13.1.jar` — Excel のソースコンポーネントとターゲットコンポーネントの機能をエミュレートします。
  - `spark-xml_2.11-0.9.0.jar`、`poi-ooxml-schemas-4.1.2.jar`、`xmlbeans-3.1.0.jar` — XML ソースコンポーネントの機能をエミュレートします。

  Python ライブラリには以下が含まれます。

  - `sct_utils.py` — ソースデータ型をエミュレートし、Spark SQL クエリのパラメータを準備します。
  - `ssis_datetime.py` — 日付と時刻の組み込み関数をエミュレートします。
  - `ssis_null.py` — ISNULL および REPLACENULL 組み込み関数をエミュレートします。
  - `ssis_string.py` — 文字列の組み込み関数をエミュレートします。

  これらのライブラリの詳細については、「AWS SCT 拡張パックにカスタムライブラリを使用する」を参照してください。

AWS SCT 拡張パックは次の 2 つの方法で適用できます。

- AWS SCT コンテキストメニューから [Apply to database] を選択してターゲットデータベーススクリプトを適用すると、拡張パックを自動的に適用できます。 AWS SCT 他のすべてのスキーマオブジェクトを適用する前に、拡張パックを適用します。
- 拡張パックを手動で適用するには、ターゲットデータベースを選択し、コンテキスト (右クリック) メニューの [拡張パックの適用]を選択します。ほとんどの場合、自動適用で十分です。ただし、パックを誤って削除した場合は、これを手動で適用する必要があります。

AWS SCT 拡張パックをターゲットデータストアに適用するたびに、コンポーネントは上書きされ、AWS SCT これに関する通知が表示されます。これらの通知をオフにするには、[設定]、[グローバル設定]、[通知] の順に選択し、[拡張パックの交換通知を非表示にする] を選択します。

Microsoft SQL Server から PostgreSQL への変換には、 AWS SCTで SQL Server から PostgreSQL への拡張パックインを使用できます。この拡張パックは SQL Server エージェントと SQL Server データベースメールをエミュレートします。詳細については、拡張パックを使用して PostgreSQL で SQL サーバーエージェントをエミュレートするおよび拡張パックを使用して PostgreSQL で SQL Server データベースメールをエミュレートするを参照してください。

AWS SCT 拡張パックの操作に関する詳細情報は以下に記載されています。

トピック

- AWS SCT 拡張パックを使用するための権限
- 拡張パックスキーマの使用
- AWS SCT 拡張パックにカスタムライブラリを使用する
- AWS LambdaAWS SCT エクステンションパックの関数を使用する
- AWS SCT 拡張パックの機能を設定する

AWS SCT 拡張パックを使用するための権限

Amazon Aurora AWS SCT の拡張パックは、関数を使用してメール送信、ジョブスケジューリング、キューイング、およびその他の操作を エミュレートします。 AWS Lambda AWS SCT 拡張パックをターゲットの Aurora データベースに適用すると、新しい AWS Identity and Access Management (IAM) ロールとインライン IAM AWS SCT ポリシーが作成されます。次に、新しい Lambda AWS SCT 関数を作成し、Aurora DB クラスターをへのアウトバウンド接続用に設定しま

す。 AWS Lambdaこれらのオペレーションを実行するには、必ず IAM ユーザーに次の必要なアクセス許可を付与してください。

- `iam:CreateRole`— アカウント用の新しい IAM ロールを作成します。 AWS
- `iam:CreatePolicy`— アカウント用の新しい IAM ポリシーを作成します。 AWS
- `iam:AttachRolePolicy`— IAM ロールに指定されたポリシーをアタッチします。
- `iam:PutRolePolicy` – IAM ロールに組み込まれているインラインポリシードキュメントを更新します。
- `iam:PassRole` — 指定した IAM ロールをルールエンジンに渡します。
- `iam:TagRole` — IAM ロールにタグを追加します。
- `iam:TagPolicy` — IAM ポリシーにタグを追加します。
- `lambda:ListFunctions` — Lambda 関数のリストを表示します。
- `lambda:ListTags` — Lambda 関数のタグのリストを表示します。
- `lambda:CreateFunction` – 新しい Lambda 関数を作成します。
- `rds:AddRoleToDBCluster` – IAM ロールを Aurora DB クラスターに関連付けます。

Amazon Redshift AWS SCT の拡張パックは、変換されたオブジェクトを Amazon Redshift に適用するときに必要なソースデータウェアハウスの基本機能をエミュレートします。変換したコードを Amazon Redshift に適用する前に、Amazon Redshift 用の拡張パックを適用する必要があります。そのためには、IAM ポリシーに `iam:SimulatePrincipalPolicy` アクションを含めてください。

AWS SCT IAM ポリシーシミュレーターを使用して、Amazon Redshift 拡張パックのインストールに必要なアクセス権限を確認します。IAM ユーザーを正しく設定していても、IAM ポリシーシミュレーターはエラーメッセージを表示することがあります。これは IAM ポリシーシミュレーターの既知の問題です。また、IAM ポリシーに `iam:SimulatePrincipalPolicy` アクションがないと、IAM ポリシーシミュレーターはエラーメッセージを表示します。このような場合は、エラーメッセージを無視して、拡張パックウィザードを使用して拡張パックを適用できます。詳細については、「拡張パックの適用」を参照してください。

拡張パックスキーマの使用

データベースまたはデータウェアハウスのスキーマを変換すると、 AWS SCT によってターゲットデータベースに別のスキーマが追加されます。この別のスキーマは、ソースデータベースの SQL システム関数を実装します。これらの関数により、変換したスキーマがターゲットデータベースに書き込まれます。この別のスキーマは、拡張パックスキーマと呼ばれます。

OLTP データベースの拡張パックスキーマの名前は、ソースデータベースに従って次の形式になります。

- Microsoft SQL Server: AWS_SQLSERVER_EXT
- MySQL: AWS_MYSQL_EXT
- Oracle: AWS_ORACLE_EXT
- PostgreSQL: AWS_POSTGRESQL_EXT

OLAP データウェアハウスアプリケーションの拡張パックスキーマの名前は、ソースデータストアに従って次の形式になります。

- Greenplum: AWS_GREENPLUM_EXT
- Microsoft SQL Server: AWS_SQLSERVER_EXT
- Netezza: AWS_NETEZZA_EXT
- Oracle: AWS_ORACLE_EXT
- Teradata: AWS_TERADATA_EXT
- Vertica: AWS_VERTICA_EXT

AWS SCT 拡張パックにカスタムライブラリを使用する

AWS SCT ソースデータベースのフィーチャをターゲットデータベースの同等の機能に変換できない場合があります。 AWS SCT 関連する拡張パックには、ターゲットデータベースの一部のソースデータベース機能をエミュレートするカスタムライブラリが含まれています。

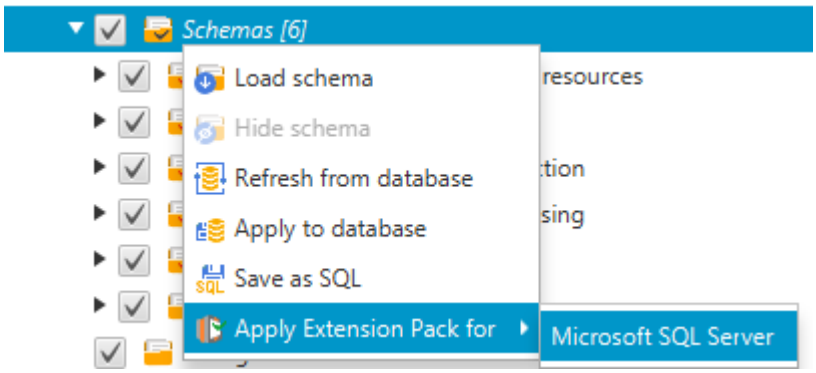
トランザクションデータベースを変換する場合は、「AWS LambdaAWS SCT エクステンションパックの関数を使用する 」を参照してください。

拡張パックの適用

エクステンションパックは、 AWS SCT エクステンションパックウィザードを使用するか、変換したコードをターゲットデータベースに適用するときに適用できます。

拡張パックウィザードを使用して拡張パックを適用するには

1. のターゲットデータベースツリーで AWS Schema Conversion Tool、コンテキスト (右クリック) メニューを開いて [Extension Pack for] を選択し、ソースデータベースプラットフォームを選択します。



   拡張パックウィザードが表示されます。

2. [Welcome] ページをお読みの上、[Next] (次へ) を選択します。
3. [AWS プロファイルの設定] ページで、以下の操作を行います。

   - 拡張パックスキーマのみを再インストールする場合は、[このステップを今すぐスキップ]、[次へ] の順に選択します。[このステップを今すぐスキップ] オプションは、オンライントランザクション処理 (OLTP) データベースでのみ使用できます。
   - 新しいライブラリをアップロードした場合は、認証情報を入力して AWS アカウントに接続します。このステップは OLAP データベースまたは ETL スクリプトを変換する場合にのみ使用してください。 AWS Command Line Interface (AWS CLI) の認証情報は、 AWS CLI がインストールされていれば使用できます。また、グローバルアプリケーション設定で以前プロファイルに保存した認証情報を使用して、プロジェクトと関連付けることもできます。必要に応じて、[グローバル設定に移動] を選択し、 AWS SCT 別のプロファイルをプロジェクトに設定または関連付けることができます。詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。

4. 新しいライブラリをアップロードする場合は、[ライブラリのアップロード] ページで [ライブラリをアップロードする必要がある] を選択します。このステップは OLAP データベースまたは ETL スクリプトを変換する場合にのみ使用してください。次に、Amazon S3 パスを指定し、[ライブラリを S3 にアップロード] を選択します。

   ライブラリを既にアップロードしている場合は、[ライブラリを既にアップロードしている] を選択し、[ライブラリのアップロード] ページで既存の S3 バケットを使用してください。次に、Amazon S3 パスを指定します。

終了したら、[Next] (次へ) を選択します。

5. [関数エミュレーション] ページで、[拡張パックの作成] を選択します。拡張パックの操作ステータスとメッセージが表示されます。

   完了したら、[Finish] (完了) を選択します。

変換したコードを適用するときに拡張パックを適用するには

1. AWS サービスプロファイルで Amazon S3 バケットを指定します。このステップは OLAP データベースまたは ETL スクリプトを変換する場合にのみ使用してください。詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。

   Amazon S3 バケットポリシーに以下のアクセス許可が含まれていることを確認してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["s3:PutObject"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["iam:SimulatePrincipalPolicy"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["iam:GetUser"],
        "Resource": ["arn:aws:iam::111122223333:user/DataExtractionAgentName"]
      }
    ]
}
```

前の例では、*111122223333:user/ DataExtractionAgentName* を IAM ユーザーの名前に置き換えてください。

2. ソースデータウェアハウススキーマを変換します。詳細については、「データウェアハウススキーマの Amazon Redshift への変換」を参照してください。
3. 右側のペインで、変換されたスキーマを選択します。
4. スキーマ要素のコンテキスト (右クリック) メニューを開き、[Apply to database] (データベースに適用) を選択します。
5. AWS SCT 必要なコンポーネントを含む拡張パックを生成し、ターゲットツリーにスキーマを追加します。aws_*database_engine_name*_ext次に、 AWS SCT 変換されたコードと拡張パックスキーマをターゲットデータウェアハウスに適用します。

   Amazon Redshift AWS Glue とをターゲットデータベースプラットフォームとして組み合わせて使用すると、 AWS SCT 拡張パックにスキーマが追加されます。

AWS LambdaAWS SCT エクステンションパックの関数を使用する

AWS SCT には、Amazon EC2 でホストされているデータベースの E メール、ジョブスケジューリング、およびその他の機能用の Lambda 関数を含む拡張パックが用意されています。

AWS Lambda 関数を使用してデータベース機能をエミュレートする

場合によって、データベース機能は同等の Amazon RDS 機能に変換できないことがあります。例えば、Oracle で送信される E メール呼び出しに UTL_SMTP が使用されている場合や、Microsoft SQL Server でジョブスケジューラが使用される場合があります。Amazon EC2 でデータベースをホストして自己管理する場合は、 AWS サービスの代わりにこれらの機能をエミュレートできます。

AWS SCT 拡張パックウィザードを使用すると、メール、ジョブスケジューリング、その他の機能をエミュレートする Lambda 関数のインストール、作成、設定を行うことができます。

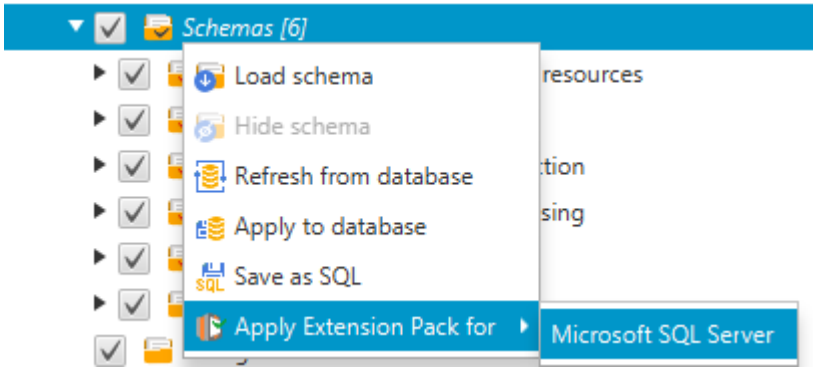
拡張パックの適用 (Lambda 関数のサポート)

拡張パックウィザードを使用するか、変換したコードをターゲットデータベースに適用するときに、拡張パックを Lambda 関数をサポートするように適用できます。

> **Important**
>
> AWS サービスエミュレーション機能は、Amazon EC2 にインストールされ、自己管理されているデータベースでのみサポートされます。ターゲットデータベースが Amazon RDS DB インスタンス上にある場合は、サービスエミュレーション機能をインストールしないでください。

拡張パックウィザードを使用して拡張パックを適用するには

1. のターゲットデータベースツリーで AWS Schema Conversion Tool、コンテキスト (右クリック) メニューを開き、[Apply extension pack for] を選択して、ソースデータベースプラットフォームを選択します。



   拡張パックウィザードが表示されます。
2. [Welcome] ページをお読みの上、[Next] (次へ) を選択します。
3. [AWS プロファイルの設定] ページで、以下の操作を行います。

   - 拡張パックスキーマのみを再インストールする場合は、[このステップを今すぐスキップ]、[次へ] の順に選択します。
   - AWS サービスをインストールする場合は、に接続するための認証情報を入力します AWS アカウント。 AWS CLI 認証情報がインストールされていれば、 AWS CLI その認証情報を使用できます。また、グローバルアプリケーション設定で以前プロファイルに保存した認証情報を使用して、プロジェクトと関連付けることもできます。必要に応じて、[Navigate to Project Settings] (プロジェクト設定に移動) を選択し、別のプロファイ次ヘルをプロジェクトに関連付けます。必要に応じて、[Global Settings] (グローバル設定) を選択して、新しいプロファイルを作成します。詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。
4. [Email Sending Service] (E メール送信サービス) ページで、以下の作業を行います。

- 拡張パックスキーマのみを再インストールする場合は、[Skip this step for now] (今のところこのステップをスキップ)、[Next] (次へ) の順に選択します。
- AWS サービスをインストールしていて、既存の Lambda 関数がある場合は、それを提供できます。それ以外の場合は、ウィザードで自動的に作成されます。終了したら、[Next] (次へ) を選択します。

5. [Job Emulation Service] (ジョブのエミュレーションサービス) で、次の作業を行います。

   - 拡張パックスキーマのみを再インストールする場合は、[Skip this step for now] (今のところこのステップをスキップ)、[Next] (次へ)の順に選択します。
   - AWS サービスをインストールしていて、既存の Lambda 関数がある場合は、それを提供できます。それ以外の場合は、ウィザードで自動的に作成されます。終了したら、[Next] (次へ) を選択します。

6. [関数エミュレーション] ページで、[拡張パックの作成] を選択します。拡張パックの操作ステータスとメッセージが表示されます。

   完了したら、[Finish] (完了) を選択します。

> **Note**
>
> 拡張パックを更新して古い拡張パックのコンポーネントを上書きするには、必ず最新バージョンのを使用してください。 AWS SCT詳細については、「のインストール、検証、更新 AWS SCT」を参照してください。

AWS SCT 拡張パックの機能を設定する

拡張パックには、使用前に設定する必要のある機能が含まれています。CONVERSION_LANG定数はサービスパックが使用する言語を定義します。関数は英語とドイツ語で使用できます。

言語を英語またはドイツ語に設定するには、関数コードを次のように変更します。次の定数宣言を探してください。

```
CONVERSION_LANG CONSTANT VARCHAR := '';
```

CONVERSION_LANG英語に設定するには、行を次のように変更します。

```
CONVERSION_LANG CONSTANT VARCHAR := 'English';
```

CONVERSION_LANG英語に設定するには、行を次のように変更します。

```
CONVERSION_LANG CONSTANT VARCHAR := 'Deutsch';
```

以下の機能にこの設定を設定します。

- aws_sqlserver_ext.conv_datetime_to_string
- aws_sqlserver_ext.conv_date_to_string
- aws_sqlserver_ext.conv_string_to_date
- aws_sqlserver_ext.conv_string_to_datetime
- aws_sqlserver_ext.conv_string_to_datetime
- aws_sqlserver_ext.parse_to_date
- aws_sqlserver_ext.parse_to_datetime
- aws_sqlserver_ext.parse_to_time

AWS SCT のベストプラクティス

AWS Schema Conversion Tool (AWS SCT) を使用するためのベストプラクティスとオプションに関する情報が見つかります。

追加メモリの構成

3,500 のストアドプロシージャを含むデータベースなどの大規模なデータベーススキーマを変換するには、AWS Schema Conversion Tool で使用可能なメモリの量を構成できます。

AWS SCT が消費するメモリの量を変更するには

1. [設定]メニューで[グローバル設定] を選択し、[JVM オプション] を選択します。
2. [設定ファイルを編集] を選択し、テキスト工ディタを選択して設定ファイルを開きます。
3. 利用可能なメモリの最小値と最大値を設定するには、JavaOptions セクションを編集します。次の例では、最小値を 4 GB に、最大値を 40 GB に設定しています。

   ```
   [JavaOptions]
   -Xmx40960M
   -Xms4096M
   ```

   使用可能な最小メモリのサイズは 4 GB 以上に設定することをお勧めします。
4. 設定ファイルを保存し、[OK] を選択し、AWS SCT を再起動して変更を適用します。

既定のプロジェクトフォルダを設定します。

AWS SCT は、プロジェクトフォルダを使用して、プロジェクトファイルの保存、評価レポートの保存、変換されたコードの保存を行います。デフォルトでは、AWS SCT はすべてのファイルをアプリケーションフォルダに保存します。別のフォルダを既定のプロジェクトフォルダとして指定できます。

既定のプロジェクトフォルダを変更するには

1. [設定] メニューで [グローバル設定] を選択し、[ファイルパス] を選択します。
2. [既定のプロジェクトファイルパス] には、既定のプロジェクトフォルダへのパスを入力します。
3. [適用]、[OK] の順に選択します。

データ移行の速度向上

1 TB を超えるデータを含む一連のテーブルなど、大きなデータセットを移行する場合は、移行速度を上げることが必要になる場合があります。データ抽出エージェントを使用する場合、データ移行の速度はさまざまな要因によって決まります。これらの要因には、ターゲット Amazon Redshift クラスターのスライスの数、移行タスクのチャンクファイルのサイズ、データ抽出エージェントを実行する PC で使用可能な RAM などが含まれます。

データ移行速度を上げるために、本番データの小さなデータセットでテスト移行セッションを複数回実行することをお勧めします。また、データ抽出エージェントは 500 GB 以上の SSD を搭載した PC で実行することをお勧めします。これらのテストセッションでは、さまざまな移行パラメーターを変更し、ディスクの使用状況を監視して、データ移行速度が最大になる構成を見つけます。次に、この構成を使用してデータセット全体を移行します。

ログ情報の増加

AWS SCT を使用して、データベース、スクリプト、アプリケーション SQL を変換する際に生成されるロギング情報を増やすことができます。ログ情報を増やすと、変換が遅くなる可能性がありますが、変更することで、エラーが発生した場合に AWS Support に確実な情報が提供されます。

AWS SCT は、ローカル環境にログを保存します。これらのログファイルを表示し、トラブルシューティングのために AWS Support または AWS SCT 開発者と共有できます。

ログ設定の変更

1. [設定] メニューを開き、[グローバル設定]、[ログ記録] を選択します。
2. [ログフォルダパス] には、ユーザーインターフェイスからログを保存するフォルダを入力します。
3. [コンソールログフォルダパス] には、AWS SCT コマンドラインインターフェイス (CLI) のログを保存するフォルダを入力します。
4. [最大ログファイルサイズ (MB)] には、1 つのログファイルのサイズ (MB) を入力します。ファイルがこの制限に達すると、AWS SCT は新しいログファイルを作成します。
5. [ログファイルの最大数] には、保存するログファイルの数を入力します。フォルダー内のログファイル数がこの制限に達すると、AWS SCT は最も古いログファイルを削除します。
6. [エクストラクタログのダウンロードパス] には、データ抽出エージェントのログを格納するフォルダを入力します。

7. [Cassandra エクストラクタログのパス] には、Apache Cassandra データ抽出エージェントのログを保存するフォルダを入力します。
8. AWS SCT がデータ抽出エージェントを使用するたびにログの保存場所を確認するようにするには、[ロードする前にパスを確認する] を選択します。
9. [デバッグモード] では、[True] を選択します。このオプションを使用すると、標準の AWS SCT ログに問題がない場合に追加情報を記録できます。
10. 主要なアプリケーションを選択して、ログ情報を増やします。以下のアプリケーションモジュールのログ情報を増やすことができます。

    - [全般]
    - [ローダー]
    - [パーサー]
    - [プリン タ]
    - [リゾルバ]
    - [Telemetry]
    - [コンバータ]
    - [型のマッピング]
    - [ユーザーインターフェイス]
    - [コントローラ]
    - [スキーマの比較]
    - [クローンデータセンター]
    - [アプリケーションアナライザ]

    前述のアプリケーションモジュールごとに、以下のログ記録レベルのいずれかを選択します。

    - [トレース] — 最も詳細な情報。
    - [デバッグ] — システム内のフローに関する詳細情報。
    - [情報] — スタートアップやシャットダウンなどのランタイムイベント。
    - [警告] — 非推奨の API の使用、API の不適切な使用、その他望ましくない、または予期しないランタイムの状況。
    - [エラー] — ランタイムエラーまたは予期しない状況。
    - [重大] — アプリケーションのシャットダウンにつながるエラー。
    - [必須] — 発生する可能性のある最高レベルのエラー。

デフォルトでは、[デバッグモード] をオンにすると、AWS SCT ですべてのアプリケーションモジュールの [情報] ログレベルが設定されます。

例えば、変換中の主な問題領域をサポートするには、[パーサー]、[型のマッピング]、および [ユーザーインターフェイス] を [トレース] に設定します。

ログがストリーミングされているファイルシステムに対して情報が冗長になりすぎた場合は、ログをキャプチャするのに十分なスペースがある場所に変更してください。

AWS Support にログをに送信するには、ログが格納されているディレクトリに移動し、すべてのファイルを管理可能な単一の .zip ファイルに圧縮します。次に、サポートケースとともに .zip ファイルをアップロードします。最初の分析が完了し、進行中の開発が再開されたら、[Debug mode] (デバッグモード) を [false] に戻して冗長なログを削除してください。その後、変換速度を上げます。

> **Tip**
>
> ログサイズを管理し、問題の報告を合理化するには、変換が成功した後にログを削除するか、別の場所に移動します。このタスクの実行により、関連するエラーと情報のみが AWS Support に送信され、ログファイルシステムがいっぱいにならないようにします。

AWS SCT で問題のトラブルシューティング

以下では、AWS Schema Conversion Tool (AWS SCT) を使用した問題のトラブルシューティングについての情報を見つけることができます。

Oracle ソースデータベースからオブジェクトをロードできない

Oracle データベースからスキーマをロードしようとしたときに、以下のいずれかのエラーが発生する場合があります。

```
Cannot load objects tree.
```

```
ORA-00942: table or view does not exist
```

これらのエラーが発生するのは、Oracle データベースへの接続に使用した ID のユーザーに、AWS SCT で必要となるスキーマを読み取るための十分なアクセス権限がないためです。

ユーザーに select_catalog_role アクセス権限を付与し、データベースのディクショナリにもアクセス権限を付与することで、この問題を解決できます。これらのアクセス権限により、AWS SCT で必要となるビューやシステムテーブルへの読み取り専用アクセスが提供されます。次の例では、min_privs という名前のユーザー ID を作成し、Oracle ソースデータベースからのスキーマの変換に必要な最小限のアクセス権限をこのユーザーに付与します。

```
create user min_privs identified by min_privs;
grant connect to min_privs;
grant select_catalog_role to min_privs;
grant select any dictionary to min_privs;
```

評価レポートの警告メッセージ

別のデータベースエンジンへの変換の複雑さを評価するには、AWS SCT がソースデータベース内のオブジェクトにアクセスできる必要があります。スキャン中に AWS SCT で問題が発生したため、評価を実行できない場合は、警告メッセージが表示されます。このメッセージは、全体のコンバージョン率が低下したことを示しています。次の理由により、スキャン中に AWS SCT に問題が発生する可能性があります。

- 必要なすべてのオブジェクトにアクセスできない。AWS SCT で必要とされるデータベースのセキュリティ許可と権限の詳細については、「AWS SCT のソース」を参照して、このガイドの適切なソースデータベースのセクションを確認してください。
- スキーマで引用されたオブジェクトがデータベース内に存在しなくなった。この問題を解決するには、SYSDBA 権限を使用して接続し、オブジェクトがデータベースに存在するかどうかを確認します。
- SCT が、暗号化されたオブジェクトを評価しようとしています。

AWS SCT CLI リファレンス

このセクションでは、 AWS SCT コマンドラインインターフェイス (CLI) を使い始める方法について説明します。また、このセクションでは、主要なコマンドと使用モードについても説明します。AWS SCT CLI コマンドの詳細なリファレンスについては、を参照してくださいリファレンス資料。

トピック

- AWS SCT コマンドラインインターフェイスを使用するための前提条件
- AWS SCT CLI インタラクティブモード
- AWS SCT CLI シナリオの取得
- AWS SCT CLI シナリオの編集
- AWS SCT CLI スクリプトモード
- AWS SCT CLI リファレンス資料

AWS SCT コマンドラインインターフェイスを使用するための前提条件

Amazon Corretto 11 の最新バージョンをダウンロードしてインストールします。ダウンロードリンクなどの詳細については、『Amazon Corretto 11 ユーザーガイド』の「Amazon Corretto 11 のダウンロード」を参照してください。

の最新バージョンをダウンロードしてインストールします。 AWS SCT詳細については、「のインストール AWS SCT」を参照してください。

AWS SCT CLI インタラクティブモード

AWS SCT コマンドラインインターフェイスはインタラクティブモードで使用できます。このモードでは、コンソールにコマンドを 1 つずつ入力します。このインタラクティブモードを使用して、CLI コマンドの詳細を確認したり、最も一般的に使用される CLI シナリオをダウンロードしたりできます。

ソースデータベーススキーマをに変換するには AWS SCT、新しいプロジェクトの作成、ソースデータベースとターゲットデータベースへの接続、マッピングルールの作成、データベースオブジェクトの変換という一連の操作を実行します。このワークフローは複雑になる可能性があるため、 AWS

SCT CLI モードではスクリプトを使用することをお勧めします。詳細については、「スクリプトモード」を参照してください。

AWS SCT CLI コマンドは、app AWS SCT インストールパスのフォルダから実行できます。Windows では、デフォルトのインストールパスは C:\Program Files\AWS Schema Conversion Tool\ です。このフォルダに AWSSchemaConversionToolBatch.jar ファイルが含まれていることを確認してください。

AWS SCT CLI インタラクティブモードに入るには、前提条件を満たした後で次のコマンドを使用します。

```
java -jar AWSSchemaConversionToolBatch.jar -type interactive
```

これで AWS SCT CLI コマンドを実行できます。/ コマンドは必ず新しい行で終了してください。また、コマンドパラメータの値の前後には必ず一重引用符 (') を使用してください。

> **Note**
>
> 前述のコマンドが Unexpected error を返す場合は、次のステップを実行します。
>
> ```
> java -Djdk.jar.maxSignatureFileSize=20000000 -jar
>  AWSSchemaConversionToolBatch.jar
> ```

AWS SCT CLI インタラクティブモードで使用できるコマンドのリストを表示するには、次のコマンドを実行します。

```
help
/
```

AWS SCT CLI コマンドに関する情報を表示するには、次のコマンドを使用します。

```
help -command: 'command_name'
/
```

前述の例では、*command_name* をコマンドの名前に置き換えます。

AWS SCT CLI コマンドのパラメータに関する情報を表示するには、次のコマンドを使用します。

```
help -command: 'command_name' -parameters: 'parameters_list'
```

```
/
```

前述の例では、*command_name* をコマンドの名前に置き換えます。次に、*parameters_list* をコンマで区切ったパラメーター名のリストに置き換えます。

AWS SCT CLI インタラクティブモードでファイルからスクリプトを実行するには、次のコマンドを使用します。

```
ExecuteFile -file: 'file_path'
/
```

前述の例では、*file_path* をスクリプトを含むファイルへのパスに置き換えます。ファイルに .scts 拡張子が付いていることを確認してください。

AWS SCT CLI インタラクティブモードを終了するには、quitコマンドを実行します。

例

Convert コマンドの出力には、次の情報が表示されます。

```
help -command: 'Convert'
/
```

次の例は、Convert コマンドの 2 つのパラメータに関する情報を表示します。

```
help -command: 'Convert' -parameters: 'filter, treePath'
/
```

AWS SCT CLI シナリオの取得

AWS SCT 最も一般的に使用されるシナリオを取得するには、GetCliScenarioコマンドを使用できます。このコマンドをインタラクティブモードで実行し、ダウンロードしたテンプレートを編集できます。編集したファイルはスクリプトモードで使用します。

GetCliScenario コマンドは、選択したテンプレートまたは使用可能なすべてのテンプレートを指定したディレクトリに保存します。テンプレートには、スクリプトを実行するためのコマンドがすべて含まれています。これらのテンプレート内のファイルパス、データベース認証情報、オブジェクト名、およびその他のデータを必ず編集してください。また、使用しないコマンドは必ず削除し、必要に応じて新しいコマンドをスクリプトに追加してください。

GetCliScenarioコマンドを実行するには、前提条件を満たし、 AWS SCT CLI インタラクティブモードに入ります。詳細については、「インタラクティブ動画」を参照してください。

次に、次の構文を使用して GetCliScenario コマンドを実行し、 AWS SCT シナリオを取得します。

```
GetCliScenario -type: 'template_type' -directory: 'file_path'
/
```

前述の例では、*template_type* を次の表のテンプレートタイプのいずれかに置き換えます。次に、*file_path* をスクリプトをダウンロードするフォルダーのパスに置き換えます。管理者権限を要求しなくてもこのフォルダにアクセスできることを確認してください。 AWS SCT また、コマンドパラメータの値の前後には必ず一重引用符 (') を使用してください。

すべての AWS SCT CLI テンプレートをダウンロードするには、-type上記のコマンドをオプションなしで実行します。

次の表には、ダウンロードできる AWS SCT CLI テンプレートのタイプが含まれています。この表には、各テンプレートのファイル名と、スクリプトを使用して実行できる操作の説明が記載されています。

テンプレートのタイプ	ファイル名	説明
BTEQ ScriptConversion	BTEQScriptConversionTemplate.scts	テラデータの基本的なメタデータクエリ (BTEQ)、FastExport FastLoad、MultiLoad およびスクリプトを Amazon Redshift RSQL に変換します。詳細については、「ETL プロセスの変換」を参照してください。
ConversionApply	ConversionTemplate.scts	ソースデータベースのスキーマを変換し、変換されたコードをターゲットデータベースに適用します。オプションで、変換したコードを SQL スクリプトとして保存し、評価

テンプレートのタイプ	ファイル名	説明
		レポートを保存します。詳細については、「データベーススキーマの変換」を参照してください。
GenericAppConversion	`GenericApplicationConversionTemplate.scts`	汎用アプリケーションコンバーターを使用して、アプリケーションに埋め込まれた SQL コードを変換します。AWS SCT 詳細については、「アプリケーション内の SQL コードの変換」を参照してください。
HadoopMigration	`HadoopMigrationTemplate.scts`	オンプレミスの Hadoop クラスターを Amazon EMR に移行します。詳細については、「Apache Hadoop を AWS SCT のソースとして使用」を参照してください。
HadoopResumeMigration	`HadoopResumeMigrationTemplate.scts`	中断されたオンプレミス Hadoop クラスターの Amazon EMR への移行を再開します。詳細については、「Apache Hadoop を AWS SCT のソースとして使用」を参照してください。

テンプレートのタイプ	ファイル名	説明
Informatica	`InformaticaConversionTemplate.scts`	Informatica の抽出、変換、ロード (ETL) スクリプトに埋め込まれている SQL コードを変換します。ETL スクリプトでソースデータベースとターゲットデータベースへの接続を設定し、変換後のスクリプトを保存します。詳細については、「Informatica ETL スクリプトの変換」を参照してください。
LanguageSpecificAppConversion	`LanguageSpecificAppConversionTemplate.scts`	C#、C++、Java、および Pro*C アプリケーションに埋め込まれた SQL コードをアプリケーションコンバータで変換します。 AWS SCT 詳細については、「アプリケーション SQL の変換」を参照してください。
OozieConversion	`OozieConversionTemplate.scts`	Apache Oozie ワークフローをに変換します。 AWS Step Functions詳細については、「Apache Oozie を AWS SCT のソースとしての使用」を参照してください。

テンプレートのタイプ	ファイル名	説明
RedshiftAgent	`DWHDataMigrationTemplate.scts`	ソースデータウェアハウスのスキーマを変換し、変換されたコードをターゲットの Amazon Redshift データベースに適用します。次に、データ抽出エージェントを登録し、データ移行タスクを作成して開始します。詳細については、「データウェアハウスから Amazon Redshift への移行」を参照してください。
ReportCreation	`ReportCreationTemplate.scts`	複数のソースデータベーススキーマのデータベース移行レポートを作成します。次に、このレポートを CSV または PDF ファイルとして保存します。詳細については、「移行評価レポート」を参照してください。
SQL ScriptConversion	`SQLScriptConversionTemplate.scts`	SQL*Plus または TSQL スクリプトを PL/SQL に変換し、変換されたスクリプトを保存します。また、評価レポートも保存します。

AWS SCT CLI テンプレートをダウンロードしたら、テキストエディタを使用して、ソースデータベースとターゲットデータベースで実行するようにスクリプトを設定します。次に、 AWS SCT CLI スクリプトモードを使用してスクリプトを実行します。詳細については、「AWS SCT CLI スクリプトモード」を参照してください。

例

次の例では、すべてのテンプレートを `C:\SCT\Templates` フォルダにダウンロードします。

```
GetCliScenario -directory: 'C:\SCT\Templates'
/
```

次の例では、ConversionApply 操作用のテンプレートを C:\SCT\Templates フォルダにダウンロードします。

```
GetCliScenario -type: 'ConversionApply' -directory: 'C:\SCT\Templates'
/
```

AWS SCT CLI シナリオの編集

シナリオテンプレートをダウンロードしたら、データベースで実行できる作業スクリプトになるように設定します。

どのテンプレートでも、ソースデータベースとターゲットデータベースのドライバーへのパスを必ず指定してください。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。

ソースデータベースとターゲットデータベースのデータベース認証情報を必ず含めてください。また、変換プロジェクトのソースとターゲットのペアを記述するマッピングルールを必ず設定してください。詳細については、「マッピングルールの作成」を参照してください。

次に、実行する操作の範囲を設定します。使用しないコマンドを削除したり、新しいコマンドをスクリプトに追加したりできます。

たとえば、ソース Oracle データベースのすべてのスキーマを PostgreSQL に変換することを計画しているとします。次に、データベース移行評価レポートを PDF として保存し、変換したコードをターゲットデータベースに適用する予定です。この場合は、ConversionApply 操作用のテンプレートを使用できます。 AWS SCT CLI テンプレートを編集するには、次の手順に従います。

ConversionApply操作用の AWS SCT CLI テンプレートを編集するには

1. ダウンロードした ConversionTemplate.scts を開きます。詳細については、「例」を参照してください。
2. -filter CreateFilter、-filter、SaveTargetSQL、ApplyToTarget SaveTargetSQ、SaveReportCSV の各オペレーションをスクリプトから削除LbyStatement、変換します。
3. オペレーション内の oracle_driver_file にはSetGlobalSettings、Oracle ドライバーへのパスを入力します。次に、[postgresql_driver_file] には、PostgreSQL ドライバーへのパスを入力します。

他のデータベースエンジンを使用する場合は、設定に適切な名前を使用してください。SetGlobalSettingsオペレーションで設定できるグローバル設定の一覧については、の「グローバル設定マトリックス」を参照してください。 リファレンス資料

4. (オプション) にはCreateProject、プロジェクトの名前とローカルプロジェクトファイルの場所を入力します。デフォルト値のまま続行する場合は、管理者権限を要求しなくても、 AWS SCT が C:\temp フォルダ内にファイルを作成できることを確認してください。
5. にはAddSource、ソースデータベースサーバーの IP アドレスを入力します。また、ソースデータベースサーバーに接続するためのユーザー名、パスワード、およびポートを入力します。
6. にはAddTarget、ターゲットデータベースサーバの IP アドレスを入力します。また、ソースデータベースサーバーに接続するためのユーザー名、パスワード、およびポートを入力します。
7. (オプション) にはAddServerMapping、マッピングルールに追加するソースデータベースオブジェクトとターゲットデータベースオブジェクトを入力します。sourceTreePath と targetTreePath パラメータを使用してデータベースオブジェクトへのパスを指定できます。オプションで、sourceNamePath および targetNamePath を使用してデータベースオブジェクトの名前を指定できます。詳細については、『リファレンス資料』の [サーバーマッピングコマンド] を参照してください。

   AddServerMappingこのオペレーションのデフォルト値は、すべてのソーススキーマをターゲットデータベースにマップします。
8. ファイルを保存し、スクリプトモードを使用して実行します。詳細については、「スクリプトモード」を参照してください。

AWS SCT CLI スクリプトモード

AWS SCT CLI スクリプトを作成するか、テンプレートを編集したら、RunSCTBatchコマンドで実行できます。必ず CLI .scts スクリプトに拡張子を付けてファイルを保存してください。

AWS SCT CLI スクリプトは、app AWS SCT インストールパスのフォルダから実行できます。Windows では、デフォルトのインストールパスは C:\Program Files \AWS Schema Conversion Tool\ です。このフォルダに RunSCTBatch.cmd または RunSCTBatch.sh ファイルが含まれていることを確認してください。また、このフォルダには AWSSchemaConversionToolBatch.jar ファイルが含まれている必要があります。

または、オペレーティングシステムの PATH 環境変数に RunSCTBatch ファイルへのパスを追加することもできます。PATH環境変数を更新すると、任意のフォルダから AWS SCT CLI スクリプトを実行できます。

AWS SCT CLI スクリプトを実行するには、Windows で次のコマンドを使用します。

```
RunSCTBatch.cmd --pathtoscts "file_path"
```

前述の例では、*file_path* をスクリプトを含むファイルへのパスに置き換えます。

AWS SCT CLI スクリプトを実行するには、Linux で次のコマンドを使用します。

```
RunSCTBatch.sh --pathtoscts "file_path"
```

前述の例では、*file_path* をスクリプトを含むファイルへのパスに置き換えます。

このコマンドには、データベースの認証情報、コンソール出力の詳細レベルなどのオプションパラメータを指定できます。詳細については、「」 AWS SCT リファレンス資料 にあるコマンドラインインターフェイスリファレンスをダウンロードしてください。

例

次の例では、C:\SCT\Templates フォルダで ConversionTemplate.scts を実行します。この例は Windows で使用できます。

```
RunSCTBatch.cmd --pathtoscts "C:\SCT\Templates\ConversionTemplate.scts"
```

次の例では /home/user/SCT/Templates ディレクトリの ConversionTemplate.scts スクリプトを実行します。この例は Linux で使用できます。

```
RunSCTBatch.sh --pathtoscts "/home/user/SCT/Templates/ConversionTemplate.scts"
```

AWS SCT CLI リファレンス資料

AWS Schema Conversion Tool コマンドラインインターフェイス (CLI) に関するリファレンス資料は、次のガイド「AWS Schema Conversion Tool CLI リファレンス」にあります。

のリリースノート AWS SCT

このセクションには AWS SCT、バージョン 1.0.640 以降の のリリースノートが含まれています。

AWS SCT ビルド 676 のリリースノート

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion Tool (SCT) の可用性	AWS DMS Schema Conversion の可用性
Oracle	PostgreL/ Aurora Postgre L	以下の関数用の新しい組み込み関数エミュレーション。 • SYS.UTL_RAW.BIT_AND(RAW,RAW) • XDB.DBMS_XSLPROCESSOR.CLOB2 FILE(CLOB) • XDB.DBMS_XSLPROCESSOR.READ2 CLOB(VARCHAR2) • SYS.UTL_RAW.BIT_OR(RAW,RAW) • SYS.UTL_RAW.BIT_COMPLEMENT(RAW)	いいえ	はい
MS SQL Server	Amazor RDS SQL Server	PDF レポートからDatabase Mail not supported メッセージを削除しました	はい	はい
Oracle	PostgreL/ Aurora Postgre L	パーティションテーブルの制約変換を実装しました。	はい	はい

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion Tool (SCT) の可用性	AWS DMS Schema Conversion の可用性
Oracle	MySQL	テーブル変換における AI-602 の適用性の確認	はい	はい
MS SQL Server	PostgreSQL/ Aurora PostgreSQL	が PostgreSQL 15.x の MERGEステートメントをサポートするようになりました	はい	はい
すべて	すべて	実装された JDBC 接続: 高度なプロパティ	はい	いいえ
すべて	すべて	CLI: PrintOLAPTaskStatus コマンドの失敗を修正	はい	いいえ
Teradat	Amazon Redshif	Teradata スタイルのデータ型キャストを実装しました。	はい	いいえ
Teradat	Amazon Redshif	SQL/BTEQ での不正なMERGE変換を修正しました。	はい	いいえ
Teradat	Amazon Redshif	Teradata スタイルのデータ型キャストを実装しました。	はい	いいえ
Teradat	Amazon Redshif	LEAD/LAG 関数変換を実装しました。	はい	いいえ
Teradat	Amazon Redshif	エラー を修正しましたAI-9996 Transformer error occurred in statement 。	はい	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion Tool (SCT) の可用性	AWS DMS Schema Conversion の可用性
Teradat	Amazor Redshif	エラー を修正しましたAI-9996 Transformer error in selectItem 。	はい	いいえ
Teradat	Amazor Redshif	部分的なストアドプロシージャの変換を実装しました。 XbiDQM.SpCmprsnDly	はい	いいえ
Teradat	Amazor Redshif	エイリアスを含む UNPIVOTステートメント。	はい	いいえ
Teradat	Amazor Redshif	複数のソーステーブルを含む Deleteステートメントを実装しました。	はい	いいえ
Teradat	Amazor Redshif	の を修正しましたAI-9996 Transformer error occurred in functionCallExpression 。	はい	いいえ
Teradat	Amazor Redshif	NORMALIZE 句変換を実装しました。	はい	いいえ
Teradat	Amazor Redshif	サブクエリを含む DELETE ステートメントの誤った変換を修正しました。	はい	いいえ
Teradat	Amazor Redshif	エラー を修正しましたAI-9996 Transformer error occurred in tableOperatorSource 。	はい	いいえ
Teradat	Amazor Redshif	エラー を修正しましたAI-9996 Transformer error occurred in additiveExpression 。	はい	いいえ
Teradat	Amazor Redshif	DBC システムオブジェクト変換を実装しました。	はい	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion Tool (SCT) の可用性	AWS DMS Schema Conversion の可用性
Teradat	Amazor Redshif	暗黙的な結合述語を使用した更新の回避策を実装しました。	はい	いいえ
Netezza	Amazor Redshif	CREATE MATERIALIZED VIEW ステートメント変換エラーを修正しました。	はい	いいえ
Db2luw	PostgreL/ Aurora PostgreL	JDBC 拡張オプション接続: 追加の接続オプションを追加しました。	はい	いいえ
Db2luw	PostgreL/ Aurora PostgreL	PostgreSQL 15.x のMERGEステートメントのサポートを追加	はい	いいえ
Db2luw	PostgreL/ Aurora PostgreL	GLOBAL TEMPORARY TABLE 変換を実装しました。	はい	いいえ
Db2luw	PostgreL/ Aurora PostgreL	USER DEFINED TYPES 変換を実装しました。	はい	いいえ
Db2luw	MySQL	GLOBAL TEMPORARY TABLE 変換を実装しました。	はい	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion Tool (SCT) の可用性	AWS DMS Schema Conversion の可用性
Db2luw	MySQL	USER DEFINED TYPES 変換を実装しました。	はい	いいえ
Db2luw	MySQL	USER DEFINED FUNCTIONS 変換を実装しました。	はい	いいえ
Db2luw	MariaDB	GLOBAL TEMPORARY TABLE 変換を実装しました。	はい	いいえ
Db2luw	MariaDB	USER DEFINED TYPES 変換を実装しました。	はい	いいえ
Sybase	すべて	Kerberos 認証のサポートを追加	はい	いいえ
Db2luw	PostgreSL/ Aurora Postgre L	ターゲットのマルチバージョン変換のサポートを追加	はい	いいえ
Azure SQL/ Microso SQL Server	PostgreSL/ Aurora Postgre L	ターゲットのマルチバージョン変換のサポートを追加	はい	いいえ
Db2luw	PostgreSL/ Aurora Postgre L	PostgreSQL 15.x の MERGEステートメントのサポートが追加されました。	はい	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion Tool (SCT) の可用性	AWS DMS Schema Conversion の可用性
Teradat	Amazor Redshif	サポートされていない関数の変更変換を修正しました。	はい	いいえ
すべて	Amazor Redshif	データエクストラクタ: インデックス付き列によるパーティショニングを実装しました。	はい	いいえ

AWS SCT ビルド 675 のリリースノート

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
Cassandr	DynamoD	ターゲットデータセンターで Cassandra のインストールが失敗するバグを修正しました。	いいえ
DB2 LUW	PostgreS(L	DYNAMIC SQL: PREPARE ステートメント: 動的 SQL なしで解決および変換します。	いいえ
DB2 LUW	PostgreS(L	REGISTER のサポートを追加しました。	いいえ
DB2 LUW	PostgreS(L	拡張パックの更新	いいえ
Hadoop	Amazon EMR	rsa-sha2 プロトコル経由で Hadoop クラスターに接続するためのサポートが追加されました。	いいえ
Microsoft SQL Server	Amazon Redshift	設定されていないにもかかわらず TLS を強制する JDBC ドライバーの修正。	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
Netezza	Amazon Redshift	マテリアライズドビュー変換のサポートが追加されました。	いいえ
Oracle	Amazon Redshift	Amazon Redshift での再帰クエリのサポートが追加されました。	はい
Oracle	PostgreSQLAurora PostgreSQL	NUMBER データ型の不正な変換を修正しました。	はい
Oracle	Amazon Redshift	データ移行。Oracle 自動パーティショニング。テーブルフラグメント値の有効期限を追加しました。有効期限は 72 時間です。有効期限が切れると、データ移行タスクの作成時にデータフラグメントが再構築されます。	いいえ
Oracle	Amazon Redshift	SCT Data Extractor: Amazon Redshift にデータをアップロードするアプローチを変更しました。デフォルトでは、エクストラクタはステージングされたテーブルを作成しません。代わりに、すべてのデータファイルが Amazon S3 バケットにあると、エクストラクタは単一の COPY コマンドを使用してターゲットテーブルにコピーします。	いいえ
Oracle	Amazon Redshift	RAW データ型の VARBYTE 列への移行を追加しました。	いいえ
Oracle	PostgreSQLAurora PostgreSQL	マルチバージョン変換	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
Oracle	PostgreSQL	PostgreSQL 15.x で MERGE ステートメントのサポートが追加されました。	はい
Oracle	PostgreSQL	PostgreSQL 15.x の新しい正規表現関数のサポートが追加されました。	はい
Oracle	PostgreSQLAurora PostgreSQL	ON CONFLICT DO UPDATE ステートメントは、除外されたエイリアスなしで変換されます。	はい
Teradata	Amazon Redshift	LEAD/LAG 関数の変換サポートを追加しました。	いいえ
Teradata	Amazon Redshift	データ形式を明示的に表示した拡張データ型キャスト。	いいえ
Teradata	Amazon Redshift	time/timestamp 式の AT 'TIME ZONE' 句の変換が改善されました。	いいえ
Teradata	Amazon Redshift	MERGE ステートメントを使用した変換手順中の AI-9996。	いいえ

AWS SCT ビルド 674 のリリースノート

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
すべて	すべて	さまざまなバグ修正とパフォーマンス向上が行われています。	一部 (サポートされているソースとターゲットのペアのみ)

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
Azure SQL/ Microsoft SQL Server	Amazon Redshift	スキーマの評価/変換中にユーザーを誤解させていた「AI 18066: スキーマ名を変換できません」というメッセージを削除	いいえ
Azure SQL/ Microsoft SQL Server	Amazon RDS for MySQL / Amazon Aurora MySQL	リターンコードを割り当てずにプロシージャを誤って変換する	一部 (スキーマ変換では現在 Azure SQL をソースとしてサポートしていません)
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	FOR XML PATH 句変換のいくつかのケースで AI9997 が修正されました。	一部 (スキーマ変換では現在 Azure SQL をソースとしてサポートしていません)
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	プロシージャ/ファンクション本体の値は元のスケールインに四捨五入されます。	一部 (スキーマ変換では現在 Azure SQL をソースとしてサポートしていません)

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	EXECUTE ステートメントの変換に関するさまざまな改善が行われました。	一部 (スキーマ変換では現在 Azure SQL をソースとしてサポートしていません)
Azure SQL/ Microsoft SQL Server/ Azure Synapse	Amazon Redshift	以下のステートメントとモードの変換が改善されました。 • EXCEPTION BLOCK • AUTOCOMMIT • NONATOMIC • GROUPING SET • CUBE • ROLLUP	いいえ
DB2 LUW	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	SQL クエリを読み込むメタデータのさまざまな修正	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
DB2 LUW	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	AI 9996 はトリガーには使用できません。	いいえ
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	行番号分析関数	いいえ
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	16 進数文字列定数のサポート	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	メタデータのさまざまな修正により、SQL クエリが読み込まれます。	いいえ
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	NEXT VALUE FOR シーケンス参照のサポート	いいえ
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	GET DIAGNOSTICS ステートメント DB2_NUMBER_ROWS オプションのサポート	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	GET DIAGNOSTICS 複数のステートメント	いいえ
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	FOR ステートメント変換のバグを修正しました。	いいえ
Oracle	Amazon RDS for MySQL / Amazon Aurora MySQL	パッケージ関数のパラメータノードが定義されていない場合のバグを修正しました。	はい
Oracle	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	エクステンションパックの関数 AWS_ORACLE_EXT.NEXT_DAY のバグを修正しました	はい

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
Oracle	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	Oracle の外部結合における変換「(+)」に関するさまざまなバグが修正されました。	はい
Oracle		Kerberos 認証をサポート	いいえ
SAP ASE	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	UPDATE ステートメントの FROM 句で複数の識別子を変換するときのバグが修正されました	いいえ
SAP ASE	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	複数行のコメントとステートメントの変換に関するバグが修正されました。	いいえ
SAP ASE		接続時の ENCRYPT_PASSWORD パラメータのサポートが追加されました。	いいえ
Teradata	Amazon Redshift	指定されたスキーマ名による VOLATILE テーブルの変換が改善されました。	いいえ

ソース	Target	新機能、強化点、修正点	AWS DMS Schema Conversion の可用性
Teradata	Amazon Redshift	複合 CTE の WHERE 句の変換が正しくありません。	いいえ
Teradata	Amazon Redshift	SCT データ抽出エージェントを使用してデータを移行する場合の INTERVAL データ型がサポートされるようになりました。	いいえ
Teradata BTEQ スクリプト	Amazon Redshift RSQL スクリプト	BTEQ によって実行されたプロシージャの変換/出力パラメータが不正です。	いいえ

AWS SCT ビルド 673 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	一般的なバグ修正とパフォーマンス向上。
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	不正な関数呼び出し変換が修正されました。
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	XML FOR 句の変換を実装しました。

ソース	Target	新機能、強化点、修正点
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	FOR XML 句のエイリアスが正しくない場合の変換。
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	AWS SCT がプロシージャパラメータを使用して文字列を実行するEXECUTEステートメントを変換しない場合のバグを修正しました。
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	内部結合による UPDATE ステートメントの変換が改善されました。
Azure Synapse	Amazon Redshift	OBJECT_ID 組み込み関数の不正な変換が修正されました。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	次のステートメントとオブジェクトの変換を実装しました。 • DECLARE TEMPORARY TABLE statement • DROP TABLE statement • PK および UNIQUE 分割テーブルに対する制約 • TIMESTAMPDIFF 関数 • TO_DATE 関数 • EBCDIC_STR 関数 • VARCHAR_FORMAT 関数

ソース	Target	新機能、強化点、修正点
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	関数ベースのインデックスが変換後に関数をスキップするバグが修正されました。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	変換後に AI 9996 で `REPEAT` ステートメントが終了するバグを修正しました。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	`FINAL TABLE` 句が 9996 で終了するバグを修正しました。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	`LOADER` | 参照制約におけるパーティショニングキー。 AWS SCT が、パーティションテーブルのプライマリキーとユニーク制約をセカンダリインデックスに変換できるようになりました。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	PostgreSQL.varchar_Format 関数のサポート

ソース	Target	新機能、強化点、修正点
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	`CreateTransformationRule` の照合順序の変更と `ModifyTransformationRule` SCT CLI コマンドを実装しました。
Greenplum	Amazon Redshift	変換後にストアドプロシージャが誤って呼び出されるバグを修正しました。
Hadoop	Amazon EMR	rsa-sha2 プロトコルを使用して Hadoop クラスターに接続するためのサポートが追加されました。
Hadoop	Amazon EMR	Glue 非対応の Hive メタストアによる Amazon EMR のサポートが追加されました。
Oracle	Amazon Redshift	PRIOR 列が SELECT リストにない再帰クエリの誤った変換に関するバグを修正しました。
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	連想配列の要素を返すように実装されました。
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	括弧で囲まれた `UNPIVOT` の予期しない AI 9996 が修正されました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	UNPIVOT での UNION ALL による予期しない AI 9996 が修正されました。
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	Number データ型変換の改善
Oracle	Amazon Redshift のデータエクストラクター	Oracle テーブルの自動パーティショニングのサポート。移行タスク作成の最適化。
Teradata	Amazon Redshift	EXCEPTION BLOCK ステートメントの変換を実装します。
Teradata	Amazon Redshift	ALL、ANY、SOMEの述語を Amazon Redshift に変換するためのサポート。
Teradata	Amazon Redshift	QUALIFY 述語のネイティブサポートが追加されました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	次の変換が改善されました。 • Recursive クエリ • `GROUPING SET` • `CUBE` • `ROLLUP` • 暗黙のジョインを含む `UPDATE` ステートメント
OLAP ソース	Amazon Redshift のデータエクストラクター	Amazon Redshift データエクストラクタータスクの停止/再開用の CLI コマンドを実装しました。
OLAP ソース	Amazon Redshift のデータエクストラクター	移行タスクの設定中に移行する必要のあるテーブル列を選択する機能が追加されました。

AWS SCT ビルド 672 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	Amazon RDS for PostgreSQL	移行対象として PostgreSQL メジャーバージョン 15 のサポートを実装しました。
すべて	Amazon Redshift	データ移行タスクのステータスを表示する新しい AWS SCT コマンドを`PrintTaskStatus` コマンドラインインターフェイス (CLI) に追加しました。

ソース	Target	新機能、強化点、修正点
すべて	Amazon Redshift	データ抽出エージェントの設定フローを改善しました。
すべて	Amazon Redshift	データ抽出エージェントがサブタスクに関する情報を表示しなかったエラーを解決しました。
Apache Oozie	AWS Step Functions	ステートマシン定義を変換後のコードにスクリプトとして保存するオプションを追加しました。
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	COALESCE、DATEADD、GETDATE、SUM 関数の変換を実装しました。
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	JOIN および OUTPUT 句を含む UPDATE ステートメントの変換が改善されました。
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SELECT TOP 1 WITH TIES ステートメントの変換中に発生したエラーを解決しました。

ソース	Target	新機能、強化点、修正点
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	組み込み関数の FOR XML 句の変換中に発生した複数の問題を解決しました。
Greenplum	Amazon Redshift	ネイティブの Amazon Redshift EXCEPTION ブロックを使用して、GET DIAGNOSTICS RAISE およびステートメントの変換を実装しました。
Greenplum	Amazon Redshift	変換されたコードに EXCEPTION ブロックのサポートを追加することで、ストアドプロシージャの変換を改善しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	時間形式テンプレートを含む TO_CHAR 関数が不適切に変換されるエラーを修正しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	ネストされたテーブル式の変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	GOTO、MERGE、REPEAT、SIGNAL ステートメントの変換を実装しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	BEFORE および AFTER方向キーワードを含む FETCH ステートメントの変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FINAL TABLE および OLD TABLEテーブル参照の変換を実装しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	以下の関数の変換を実装しました。 • ADD_MONTHS • 文字データ型のパラメータを持つ DAY • DAYOFWEEK • DAYS • DECODE • HOUR • LAST_DAY • LOCATE_IN_STRING • MICROSECOND • MINUTE • MONTH • ROUND • TIME • TIMESTAMP • TIMESTAMP_FORMAT • TRANSLATE • UNICODE_STR • XMLCAST • XMLELEMENT • XMLQUERY • XMLSERIALIZE • YEAR
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	JOIN 句内のサブクエリーのエイリアスの変換が改善されました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	COALESCE 関数の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	EXPLICIT インデックスの変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	複合式の列名の変換を改善し、変換中にアクション項目 9997 が予期せず表示される問題を解決しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	プライマリキーとユニーク制約の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	INSERT ステートメント内の XMLTABLE ステートメントの変換を改善し、変換中にアクション項目 9996 が予期せず表示される問題を解決しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SUBSTR 引数を持つ関数の変換中にアクション項目 9996 が予期せず表示される問題を解決しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CURRENT TIMESTAMP 特別レジスターの変換中にアクション項目 9996 が予期せず表示される問題を解決しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	ステートメント、サポートされていないステートメント、およびサポートされていない組み込み関数の変換中にアクション アイテム 9996 が予期せず表示される問題を解決しました。
Microsoft SQL Server	すべて	ソースとして Microsoft SQL Server バージョン 2022 のサポートを追加しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	文字列連結演算子を使用するSELECTステートメントの変換が改善されました。変換されたコードで STRING_AGG 関数 AWS SCT を使用します。
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish 機能設定ファイルの新バージョン 3.1.0 のサポートを実装しました。このファイルには、特定の Babelfish バージョンでサポートされているものとサポートされていない SQL 機能が定義されています。

ソース	Target	新機能、強化点、修正点
Netezza	Amazon Redshift	データ抽出エージェントが指定の CDC ポイントからデータ移行を開始しなかった問題を解決しました。
Oracle	すべて	Oracle データベースバージョン 19 の評価レポートをソースとして更新しました。
Oracle	Aurora PostgreSQL PostgreSQL	AWS SCT 拡張パックに新しい関数を追加してDBMS_OUTPUT 、パッケージの変換を実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	連想配列を引数またはパラメータとして使用する関数とプロシージャの変換を実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	SELECT ステートメントの DISTINCT 句の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	プライマリキー制約の名前がテーブルと同じであるテーブルの変換が改善されました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	3 番目のパラメータによる RAISE_APPLICATION_ERROR プロシージャの変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	移行ルールによって NUMERIC データ型が該当する場所 INTEGER に自動的に変更されなかった問題を解決しました。
Oracle DW	Amazon Redshift	変換されたコードにネイティブ Amazon Redshift CONNECT BY 句のサポートを実装しました。
Oracle DW	Amazon Redshift	移行スコープ内の各テーブルまたはパーティションにサブタスクを自動的に追加することで、データ移行を改善しました。このアプローチにより、パーティショニング後に挿入されたデータのデータが失われるのを防ぐことができます。
Teradata	Amazon Redshift	再帰的ビューの変換を実装しました。
Teradata	Amazon Redshift	ネイティブの Amazon Redshift AUTOCOMMIT トランザクションモードのサポートを追加することで、BTET および ANSI トランザクションモードを使用するストアドプロシージャの変換を改善しました。
Teradata	Amazon Redshift	変換されたコードに NONATOMIC キーワードを追加することで、TERADATA トランザクションセマンティックを使用するストアドプロシージャの変換が改善されました。
Teradata	Amazon Redshift RSQL	変換されたコードに AWS アクセスキー ID とシークレットアクセスキーが含まれている問題を解決しました。

AWS SCT ビルド 671 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	に Windows にプロジェクトファイルを保存するアクセス許可 AWS SCT がないというエラーを修正しました。
すべて	すべて	次の AWS SCT コマンドラインインターフェイス (CLI) テンプレートを更新しました。 • BTEQScriptConversion • ConversionApply • HadoopMigration • HadoopResume移行 • Informatica AWS SCT CLI テンプレートの詳細については、「」を参照してくださいCLI シナリオの取得。
すべて	Amazon Redshift	がコマンドラインインターフェイス (CLI) で拡張パックを作成しないというエラーを修正 AWS SCT しました。
すべて	Amazon Redshift	AWS SCT データ抽出エージェントがコマンドラインインターフェイス (CLI) で AWS Snowball 設定を使用しなかった問題を解決しました。
Apache Oozie	AWS Step Functions	AWS Step Functions コマンドラインインターフェイス (CLI) モードで Apache Oozie から への移行のサポートを実装しました。Hadoop ワークロードを Amazon EMR に移行したら、ワークフロースケジューリングシステムを AWS クラウドに移行できるようになりました。詳細については、「Apache Oozie を AWS Step Functions に変換 する」を参照してください。
Azure SQL データベース	Aurora PostgreSQL	テーブルとエイリアスで発生したリゾルバーエラーを修正しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	PostgreSQL	
Azure SQL データベース	Aurora PostgreSQL	INDEX ON 句の変換を実装しました。
Microsoft SQL Server	PostgreSQL	
Azure SQL データベース	Aurora PostgreSQL	予期しないアクション項目を回避するために、次のオブジェクトの変換を改善しました。 • Batch ステートメント • 式のリスト • テーブルエイリアス • 一時テーブル • トリガー • ユーザー変数
Microsoft SQL Server	PostgreSQL	
Azure SQL データベース	Aurora PostgreSQL	プロシージャで発生した解析エラーを解決しました。
Microsoft SQL Server	PostgreSQL	

ソース	Target	新機能、強化点、修正点
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	OBJECT_ID 関数の変換されたコードで一時テーブルの名前が正しく AWS SCT ないというエラーを修正しました。
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	以下のコード要素の変換中にアクション項目 9996 が予期せず表示される問題を解決しました。 • CONVERT 関数 • DATEADD 関数 • インライン関数内の DELETE ステートメント • IF ステートメント • カラムの INSERT または UPDATE アクション • RETURN ステートメント • 複雑なクエリや関数を含む UPDATE ステートメント
BigQuery	Amazon Redshift	マルチサーバー評価プロセスのソース BigQuery として のサポートを追加しました。詳細については、「マルチサーバー評価レポートの作成」を参照してください。
Hadoop	Amazon EMR	ソースデータベースへの接続に使用する、サポートされている Apache Hive JDBC ドライバーのバージョンを更新しました。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	ソースメタデータローダーを強化して、 がプライマリキー、暗黙的なインデックスなどのソースデータベースオブジェクトを AWS SCT ロードすることを確認します。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	暗黙カーソルの列で発生したリゾルバーエラーを修正しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	変換されたコードの DML ステートメントの列名、式、句のフォーマットを保持する機能が実装されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	クロススキーマの外部キーの変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	`LENGTH` および `VARCHAR` 関数の変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	`LABEL ON` および `DECLARE CONDITION` ステートメントの変換を実装しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	OPTIMIZE FOR 句を含む SELECT ステートメントの変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	サポートされているすべてのデータ型にデフォルト値を追加することで、CREATE TABLE ステートメントの変換を改善しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	INCREMENT BY 属性の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	テーブルパーティションを変換スコープから除外する機能が追加され、分割テーブルの変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	プライマリキー定義と INCLUDE 列の変換が改善されました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SUBSTRING 関数の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SET および DECLARE HANDLER FOR ステートメントの変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	変数データ型の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	XMLTABLE 関数の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	変換されたオブジェクトをターゲットデータベースに適用する順序 (テーブル、パーティション、インデックス、制約、外部キー、トリガー) を実装することで、移行フローを改善しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	ソースコード内のコメントの変換中にアクション項目 9996 が予期せず表示される問題を解決しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FROM 句内の エイリアスの変換中にアクション項目 9997 が予期せず表示される問題を解決しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	カーソル エイリアスの変換中にアクション項目 9997 が予期せず表示される問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	変換されたコードが、ORDER BY 句を含む SELECT ステートメントに対して異なる結果を返すというエラーを修正しました。SQL Server と PostgreSQL では NULL 値の扱いが異なるため、変換後のコードには、変換後のコードがソースコードと同じ順序で結果を返すようにするための NULLS FIRST または NULLS LAST 句が含まれるようになりました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	テーブル関数のデータ型が不適切に変換される問題を解決しました。

ソース	Target	新機能、強化点、修正点
MySQL	Amazon RDS for MySQL	変換されたコードのデータベースオブジェクト名の前後に一重引用符 (' ') が予期せず表示される問題を解決しました。
Oracle	Aurora PostgreSQL PostgreSQL	パーティションとサブパーティションに関する情報を表示するために使用する Oracle システムビューをエミュレートする新しいビューを拡張パックに追加しました。
Oracle	Aurora PostgreSQL PostgreSQL	拡張パックの 2 つの関数が更新され、変換されたコードの引数としてスキーマ名が追加されました。
Oracle	Aurora PostgreSQL PostgreSQL	ユーザーインターフェイスでアプリケーションコードを更新した後、 AWS SCT が C++ アプリケーションの変換に正しいパラメータを使用しなかったエラーを修正しました。
Oracle	Aurora PostgreSQL PostgreSQL	予期しない例外が発生しないように、`CREATE TYPE` ステートメントの変換を改善しました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	ネストされたテーブルの変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	パッケージオブジェクトで発生した解析エラーを解決しました。
Oracle	Aurora PostgreSQL PostgreSQL	名前の長さが 60 文字を超えると、変換されたコード内のオブジェクト名が AWS SCT 予期せずトリミングされる問題を解決しました。
Oracle	Aurora PostgreSQL PostgreSQL	分割テーブルの行レベルのトリガーが不適切に変換される問題を解決しました。
Oracle DW	Amazon Redshift	データ移行のための自動テーブルパーティショニングのサポートを実装しました。データ移行を高速化するために、 `ROWID` は擬似列の値に基づいて大きなテーブルまたはパーティションを自動的にパーティション分割 AWS SCT できます。詳細については、「ネイティブパーティショニングを使用する」を参照してください。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	変換された Amazon Redshift コードにネイティブ MERGE コマンドのサポートを実装しました。Amazon Redshift の MERGE コマンドの詳細については、『Amazon Redshift データベースデベロッパーガイド』の「移行」を参照してください。
Teradata	Amazon Redshift	明示的なテーブル名を使用しない DELETE および UPDATE ステートメントの変換が改善されました。
Teradata	Amazon Redshift	IN および NOT IN ステートメントが誤って変換される問題を解決しました。

AWS SCT ビルド 670 のリリースノート

ソース	Target	新機能、強化点、修正点
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	以下のコード要素の変換中にアクション項目 9996 が予期せず表示される問題を解決しました。 • INCLUDE ステートメント内の CREATE INDEX ステートメント • DECLARE ステートメント • DECLARE ... TABLE ステートメント • LOOP ステートメントの内部にデフォルト値を持つ DECLARE を含む • DELETE ステートメント • ALTER TABLE ステートメント内の DROP CONSTRAINT ステートメント • EXECUTE AS CALLER および REVERT • IIF ステートメント • 式のリスト • MONTH( ) 関数 • UPDATE ステートメント • YEAR( ) 関数

ソース	Target	新機能、強化点、修正点
Azure Synapse Analytics	Amazon Redshift	マルチサーバー評価プロセスのソースとしての Azure Synapse Analytics に対するサポートが追加されました。詳細については、「マルチサーバー評価レポートの作成」を参照してください。
Hadoop	Amazon EMR	コマンドラインインターフェイス (CLI) モードで Hadoop クラスターを Amazon EMR に移行するためのサポートを実装しました。詳細については、「ビッグデータフレームワークの移行」を参照してください。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	ソーステーブルと列で発生したリゾルバーエラーを修正しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CASE 式の変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CURRENT_DATE 参照の特殊レジスターへの変換を実装しました。Db2 for z/OS の特殊レジスターへの参照は、現在のサーバーから提供された値への参照です。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	DATE および POSSTR 関数の変換を実装しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	日時定数の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	DATE、TIME、TIMESTAMP 、TIMESTAMP WITH TIME ZONE およびの各データ型の列のデフォルト値の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SELECT INTO ステートメントの変換中にアクション項目 9996 が予期せず表示される問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	DATEDIFF 関数の変換が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	ISNULL 関数が NULLIF に変換される際のエラーを修正しました。その結果、変換されたコードはソースコードとは異なる結果になりました。これで、 はISNULL関数を AWS SCT に変換しますCOALESCE。
Netezza	Amazon Redshift	データ抽出エージェントが改良され、正常に完了したタスクに障害ステータスが設定されていた問題を解決しました。

ソース	Target	新機能、強化点、修正点
Netezza	Amazon Redshift	データ抽出エージェントによるデータ移行を開始した後に、サブタスク内のエンドポイントを変更する機能が追加されました。
Microsoft SQL Server MySQL Oracle PostgreSQL	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	IPv6 アドレスプロトコルを使用してデータベースに接続する機能が追加されました。
Oracle	「Amazon RDS for Oracle」	ジョブキュー内のジョブをスケジュールして管理する DBMS_JOB パッケージの変換を実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	グローバルネストテーブルの変換を改善する新しい関数を拡張パックに追加しました。これらの新しい関数は、Oracle のソースコード内のDELETE、EXTEND、TRIM 関数をエミュレートします。
Oracle	Aurora PostgreSQL PostgreSQL	Java アプリケーションに埋め込まれている SQL コードの変換範囲を指定する機能が追加されました。ソースアプリケーションプロジェクトのサブセットを変換スコープから除外できるようになりました。詳細については、「AWS SCT での Java アプリケーションの SQL コードの変換」を参照してください。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	機能インデックス内の連結演算子 (||) の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	ソースコードに 1 つの式を表す括弧が含まれていない IN 条件の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	PostgreSQL での MERGE ステートメントの INSERT ON CONFLICT への変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	プロシージャのパッケージで発生した解析エラーを解決しました。
Oracle	Aurora PostgreSQL PostgreSQL	パッケージの変換中にアクション項目 5072 が予期せず表示される問題を解決しました。
Oracle DW	Amazon Redshift	変換されたコードをターゲットデータベースに適用する際に、 が拡張パックを適用しなかったエラーを修正 AWS SCT しました。

ソース	Target	新機能、強化点、修正点
Oracle DW	Amazon Redshift	拡張パックウィザードの使用時に、 が一部の拡張パックファイルを適用しなかったエラーを修正 AWS SCT しました。
Oracle DW	Amazon Redshift	500 AWS SCT を超えるタスクが並行して実行されている が への AWS Snowball データ移行を処理できない問題を解決しました。
Oracle DW	Amazon Redshift	ユーザー定義型を持つユーザー定義関数が不適切に変換される問題を解決しました。

AWS SCT ビルド 669 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	ソースデータベースに最適なターゲットデータベースプラットフォームを決定するのに役立つマルチサーバー評価プロセスを改善しました。ここで、入力カンマ区切り値 (CSV) ファイルでデータベース認証情報を指定すると、 はキー AWS SCT を無視します AWS Secrets Manager 。詳細については、「マルチサーバー評価レポートの作成」を参照してください。
すべて	すべて	からのシークレットを使用してデータベース AWS Secrets Manager に接続するときに、マルチサーバー評価レポートにソースデータベースの IP アドレスが含まれている問題を解決しました。
すべて	Amazon Redshift	オペレーティングシステムと使用可能な RAM に応じて Java 仮想マシン (JVM) 設定の自動設定を実装しました。 はこの JVM AWS SCT を使用してデータ抽出エージェントを実行します。
すべて	Amazon Redshift	Ubuntu でデータ抽出エージェントが起動しない問題を解決しました。
すべて	Amazon Redshift	Windows で `StartAgent.bat` ファイルを実行してもデータ抽出タスクが開始されない問題を解決しました。

ソース	Target	新機能、強化点、修正点
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	[インデックスの固有の名前を生成] オプションがオンになっていると、列名が不適切に変換される問題を解決しました。
Greenplum	Amazon Redshift	VOID がプロシージャに戻る関数の変換を実装しました。
Greenplum	Amazon Redshift	ソースデータベースに数値列に数値 (NaN) 値が含まれていない場合にデータ移行が失敗する問題を解決しました。 AWS SCT データ抽出エージェントは NaN 値を NULL に置き換えるようになりました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CHAR 組み込み関数の変換中に DATE FORMAT および TIME FORMAT オプションを指定する新しい変換設定が追加されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	WITHOUT RETURN 句で宣言された定義済みカーソルを変換するためのアクション項目 8534 が追加されました。カーソルが結果セットを返さない場合、 は変換されたコードのカーソル名にNULL値を AWS SCT 割り当て、アクション項目を生成します。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	ソースデータベースへの接続中に AWS SCT で識別される CURRENT CLIENT_APPLNAME プロパティを編集しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CHAR 組み込み関数の変換中に DATE FORMAT および TIME FORMAT オプションを指定する新しい変換設定を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	BEGIN...END ブロックステートメント内の LEAVE ステートメントの変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	XMLPARSE、XMLTABLE、XMLNAMESPACES 関数の変換を実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CHAR 組み込み関数の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	カーソルの変換が改善されました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FOR ループステートメントの変換中にアクション項目 9996 が予期せず表示される問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SELECT ステートメント内でのテーブルタイプの使用方法の変換が改善されました。
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish 機能設定ファイルの新バージョン 2.2.0 のサポートを実装しました。このファイルには、特定の Babelfish バージョンでサポートされているものとサポートされていない SQL 機能が定義されています。
Netezza	Amazon Redshift	データ抽出エージェントが改良され、進行中のデータ複製中にターゲットテーブルから 1 行が削除されないという問題を解決しました。
Oracle	「Amazon RDS for Oracle」	Oracle Database Enterprise Edition 機能の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	GROUPING_ID 関数の変換を実装しました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	コマンドラインインターフェイス (CLI) モードでのカスタムデータ型マッピングのサポートを追加することで、C# アプリケーションの SQL コード変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	予期しないアクション項目 9996 が発生しないように、ネストされたテーブルの変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	オブジェクトコンストラクターの呼び出しが不適切に変換される問題を解決しました。
Oracle DW	Amazon Redshift	データ移行用の既存のテーブルパーティションのサポートを実装しました。データ移行を高速化するために、 は、空でないソーステーブルのパーティションごとにサブタスク AWS SCT を作成します。詳細については、「ネイティブパーティショニングを使用する」を参照してください。
Teradata	Amazon Redshift	`TIME WITH TIME ZONE AS TIMESTAMP` 、`TIME WITH TIME ZONE AS CHAR`、`TIMESTAMP AS TIME WITH TIME ZONE` 引数を含む `CAST` 関数の変換が改善されました。
Teradata	Amazon Redshift	`FORMAT` オプションによる `CAST` 関数の変換が改善されました。
Teradata	Amazon Redshift	`CEIL` 関数が変換されない問題を解決しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	DELETE 句を含む MERGE ステートメントが不適切に変換される問題を解決しました。
Teradata	Amazon Redshift	日付とフォーマットの引数を持つ TO_CHAR 関数が不適切に変換される問題を解決しました。

AWS SCT ビルド 668 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	Amazon Redshift	マイグレーションルール内の乗算演算子が正しく動作しない問題を解決しました。これらの演算子により、char、varchar、nvarchar、string データ型の長さを変更できます。詳細については、「移行ルールの作成」を参照してください。
Azure Synapse Analytics	Amazon Redshift	VARCHAR 引数付き CONVERT 関数のサポートを実装しました。
Azure Synapse Analytics	Amazon Redshift	NOLOCK 句を含む SELECT ステートメントの変換が改善されました。
Azure Synapse Analytics	Amazon Redshift	エイリアスまたは SET および FROM 句を含む UPDATE ステートメントの変換が改善されました。
Greenplum	Amazon Redshift	データ移行用の自動仮想パーティショニングを実装しました。 AWS SCT は GP_SEGMENT_ID システム列を使用してパーティションを作成します。
Greenplum	Amazon Redshift	RETURN QUERY および RETURN SETOF 句のサポートを実装しました。

ソース	Target	新機能、強化点、修正点
Greenplum	Amazon Redshift	3 つのパラメータを持つ SUBSTRING 関数のサポートを実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	LOCATEパラメータによる SUBSTR 関数の変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	REFCURSOR 変数の配列を使用して動的結果セットを返すオプションが追加されました。変換設定でこのオプションを選択すると、AWS SCT で変換されたコードに OUT パラメータが追加されます。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FOR ループステートメントのサポートを実装しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	XMLPARSE 関数のサポートを実装しました。XMLPARSE 関数内のスペーストライピングのアクション項目 8541 を追加しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	1 つの BEGIN ... END ブロック内の複数の例外ハンドラーの変換が改善されました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	INSERT および DELETE トリガーの変換が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	ネストされたプロシージャコールの変換が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	テーブル型の変換が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	ビット単位の論理 NOT 演算が整数値に誤って変換される問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	PostgreSQL バージョン 8.0.2 以前でローカル配列が初期化されない問題を解決しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	WHEN NOT MATCHED BY SOURCE 句を含む MERGE ステートメントが不適切に変換される問題を解決しました。
MySQL	Aurora MySQL	rds_superuser_role ロールによって付与されたユーザーアクセス許可を AWS SCT 誤って判断する問題を解決しました。
Netezza	Amazon Redshift	ソースメタデータローダーを強化して、 が名前が小文字のデータベースオブジェクト AWS SCT を正しくロードするようにしました。
Oracle	Aurora PostgreSQL PostgreSQL	ローカルのネストテーブルの変換を改善する 新しい関数が拡張パックに追加されました。これらの新しい関数は、ソース Oracle コード内のPRIOR、NEXT、LIMIT、FIRST、LAST、EXISTS、EXTEND、TRIM、DELE および SET 関数をエミュレートします。詳細については、「拡張パックの使用」を参照してください。
Oracle	Aurora PostgreSQL PostgreSQL	C# アプリケーションの変換スコープを指定する機能が追加されました。ユーザーは、ソースアプリケーションプロジェクトのサブセットを変換スコープから除外できるようになりました。
Oracle	Aurora PostgreSQL PostgreSQL	コレクション内の COUNT メソッドのサポートを実装しました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	ネストされたテーブル内の変数とコンストラクターのサポートを実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	`RATIO_TO_REPORT` および `STANDARD_HASH` 関数のサポートを実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	AWS SCT 拡張パックの一部としてのラージオブジェクト (LOB) の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	ローカルコレクションの変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	列名にテーブル名が含まれていない `USING` 句を含む `JOIN` ステートメントの変換が改善されました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	EMPTY_BLOB および EMPTY_CLOB 関数の変換を実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	C# アプリケーションでの位置バインド変数の変換を実装しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	マルチイベントトリガーの変換を実装しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	再帰トリガーの変換を実装しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	@@rowcount グローバル変数によるトリガーの変換が改善されました。

ソース	Target	新機能、強化点、修正点
SAP ASE	Aurora PostgreSQL PostgreSQL	UPDATE ステートメントの SET 句内の集計関数が誤って変換される問題を解決しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	UPDATE ステートメントの変換中にアクション項目 42702 が予期せず表示される問題を解決しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	CHAR 引数のある CONVERT 関数が不適切に変換される問題を解決しました。
Snowflake	Amazon Redshift	データ抽出エージェントによるデータ移行のソースとしての Snowflake AWS SCT のサポートが追加されました。詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。
Teradata	Amazon Redshift	TIMESTAMP AS TIME WITH TIMEZONE 引数付き CAST 関数の変換が改善されました。

AWS SCT ビルド 667 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	コマンドラインインターフェイス (CLI) モードで Informatica 抽出、変換、ロード (ETL) スクリプトのサポートを実装しました。Informatica ETL スクリプトは、新しいターゲットデータベース

ソース	Target	新機能、強化点、修正点
		AWS SCT に自動的にリダイレクトされます。また、 は Informatica オブジェクトに埋め込まれているオブジェクト名と SQL コードを AWS SCT 変換します。詳細については、「Informatica ETL スクリプトの変換」を参照してください。
すべて	Amazon Redshift	Amazon Redshift でサポートされる最小ドライバーバージョンが 2.1.0.9 になりました。詳細については、「必要なデータベースドライバーのダウンロード」を参照してください。
Azure Synapse Analytics	Amazon Redshift	3 つの日付と時刻の引数を持つ CONVERT 関数の変換を改善する新しい関数を拡張パックに追加しました。
Azure Synapse Analytics	Amazon Redshift	DATEDIFF 関数の変換が改善されました。
Azure Synapse Analytics Microsoft SQL Server DW	Amazon Redshift	拡張パックのバージョンを更新しました。拡張機能パックの最新バージョンを既存の AWS SCT プロジェクトに適用してください。詳細については、「拡張パックの使用」を参照してください。
BigQuery	Amazon Redshift	フィルタリングされたオブジェクトがコマンドラインインターフェイス (CLI) モードで変換されない問題を解決しました。
Greenplum	Amazon Redshift	がストア AWS SCT ドプロシージャで宣言された一時テーブルを変換しないというエラーを修正しました。
Greenplum	Amazon Redshift	変換されたコードで列エンコーディング属性が欠落していたエラーを修正しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	複数の INNER JOIN 句を含む自己参照テーブルの UPDATE ステートメントの変換を実装しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SQL Server が DML トリガーに使用する inserted および deleted テンポラリテーブルのサポートとテンポラリテーブルを実装しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	さまざまなデータベーススキーマで作成されたストアドプロシージャ内のユーザー定義型の変換が改善されました。がデータ型を見つけられず、アクション項目 AWS SCT 9996 が表示される問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	変換されたコード内のデータベースオブジェクト名の前後に角括弧 ([ ]) が予期せず表示される問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	@@ROWCOUNT 関数が誤って変換される問題を解決しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server DW	Amazon Redshift	`geometry` および `geography` データ型のサポートを実装しました。
Microsoft SQL Server DW	Amazon Redshift	変換後のコードのデータ型宣言における `MAX` キーワードのサポートを実装しました。
Microsoft SQL Server DW	Amazon Redshift	`DATEADD` 関数の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	フレームワークのサポートを追加することで、Java アプリケーションの SQL コード変換が改善されました MyBatis。詳細については、「Java アプリケーションの SQL コードの変換」を参照してください。
Oracle	Aurora PostgreSQL PostgreSQL	MyBatis フレームワークを使用する Java アプリケーションの SQL コード変換が改善されました。サポートされていない構文の SQL コード用のアクション項目 30411 を追加しました。
Oracle	Aurora PostgreSQL PostgreSQL	`typedef struct` 宣言のサポートを追加することにより、Pro*C アプリケーションでの SQL コード変換が改善されました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	CROSS JOIN および LEFT JOIN ステートメントのサポートを実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	MERGE ステートメントの変換が向上しました。変換されたコードに挿入する値が欠落していた問題を解決しました。
Teradata	Amazon Redshift	変換されたコードで AWS SCT が使用するデフォルトの列圧縮エンコード設定を、デフォルトの Amazon Redshift 設定と一致するように変更しました。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「圧縮エンコード」を参照してください。
Teradata	Amazon Redshift	TIME データ型を使用する数学演算が不適切に変換される問題を解決しました。
Teradata	Amazon Redshift RSQL	シェルスクリプト内の FastExport コードの変換を実装しました。
Teradata BTEQ	Amazon Redshift RSQL	が COALESCEおよび %dataステートメントを変換しないというエラー AWS SCT を修正しました。
Vertica	Amazon Redshift	ユーザーが 1 つの最適化戦略を選択した場合の変換最適化の提案が改善されました。

AWS SCT ビルド 666 のリリースノート

ソース	Target	新機能、強化点、修正点
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	JOIN ステートメントの内部にある ON 句で発生する解析エラーを解決しました。
Azure Synapse Analytics	Amazon Redshift	日付と時刻の引数を持つ関数の変換を改善するため、拡張パックに 3 つの新しい CONVERT 関数が追加されました。
Azure Synapse Analytics	Amazon Redshift	がシステムデータベーススキーマを AWS SCT ロードするようにソースメタデータローダーを強化しました。
Azure Synapse Analytics	Amazon Redshift	テンポラリテーブルの列で発生したリゾルバーエラーを修正しました。
Azure Synapse Analytics	Amazon Redshift	BINARY および VARBINARY データ型から VARBYTE データ型への変換を実装しました。
Azure Synapse Analytics	Amazon Redshift	変換されたコード内の TIME データ型のサポートを実装しました。
Azure Synapse Analytics	Amazon Redshift	COLLATE 句の変換が改善されました。デフォルトのデータベース照合順序で列を変換すると、アクション項目 31141 が予期せず表示される問題を解決しました。
BigQuery	Amazon Redshift	入力パラメータを変更するプロシージャの変換を実装しました。

ソース	Target	新機能、強化点、修正点
Greenplum	Amazon Redshift	が Greenplum 6.x データベースと互換性のないクエリ AWS SCT を使用した問題を解決しました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	Db2 for z/OS から例外ハンドラーを PostgreSQL に転送することで、EXCEPTION セクションの変換が改善されました。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	OPEN CURSOR ステートメントの変換が向上しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	CASE 式を使用した IIF 関数の変換を実装しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	CREATE PROCEDURE ステートメントに BEGIN...END ブロックが含まれていない場合に、テーブル値パラメーターを含むプロシージャが誤って変換される問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SCOPE_IDENTITY 関数が誤って変換される問題を解決しました。

ソース	Target	新機能、強化点、修正点
Oracle	「Amazon RDS for Oracle」	Oracle 10g をソースとして使用しているときに SELECT_CATALOG_ROLE ロールで発生するローダーエラーを修正しました。
Oracle	「Amazon RDS for Oracle」	Oracle スケジューラジョブをサポートするようにローダーを改善しました。
Oracle	Aurora PostgreSQL PostgreSQL	USING 句を含む JOIN ステートメントの変換を実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	ソースコードの WHERE 句にグローバル変数が含まれる変換コードのパフォーマンスが向上しました。
Oracle	Aurora PostgreSQL PostgreSQL	フレームワークのサポートを追加することで、Java アプリケーションの SQL コード変換が改善されました MyBatis。詳細については、「Java アプリケーションの SQL コードの変換」を参照してください。
Oracle DW	Amazon Redshift	PIVOT および UNPIVOT 関係演算子の変換を実装しました。
Teradata	Amazon Redshift	JSON オブジェクトを使用するソースコードが変換されなかったエラーを修正しました。
Teradata	Amazon Redshift	ドロップしたユーザーが作成したテーブルが正しく読み込まれなかったエラーを修正しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	INSTR 関数からネイティブの Amazon Redshift STRPOS 関数への変換を実装しました。
Teradata	Amazon Redshift	NVP および TRANSLATE 関数の変換を実装しました。
Teradata	Amazon Redshift	COALESCE 式の変換が改善されました。
Teradata	Amazon Redshift	DECLARE CONDITION ステートメントの変換が向上しました。
Teradata	Amazon Redshift	SECOND 構文要素を含む EXTRACT 関数の変換が改善されました。
Teradata	Amazon Redshift	LOOP ステートメント内の SQLSTATE および SQLCODE 変数の変換が改善されました。
Teradata	Amazon Redshift	ユニークインデックスの変換が改善されました。
Teradata	Amazon Redshift	小数精度が 3 に設定されている CURRENT_TIMESTAMP ステートメントの変換中にアクション項目 9996 が予期せず表示される問題を解決しました。
Teradata	Amazon Redshift	文字列リテラルでバックスラッシュが誤って変換される問題を解決しました。
Teradata	Amazon Redshift	変換された EXEC ステートメントの ADD CONSTRAINT ステートメントに誤ったフィールド名が含まれていた問題を解決しました。
Teradata	Amazon Redshift	変換された QUALIFY サブクエリに誤ったサブクエリ名が含まれていた問題を解決しました。
Teradata	Amazon Redshift	変換されたビューが適用されない問題を解決しました。変換されたコードの NULL 値を特定のデータ型に明示的にキャストできるようになりました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	日付と時刻の関数が不適切に変換される問題を解決しました。
Teradata	Amazon Redshift	16 進数の文字列リテラルが変換されない問題を解決しました。

AWS SCT ビルド 665 のリリースノート

ソース	Target	新機能、強化点、修正点
Azure Synapse Analytics	Amazon Redshift	VARCHAR 引数を持つ CONCAT 関数の変換を実装しました。
Azure Synapse Analytics	Amazon Redshift	一時テーブルを作成し、スキーマ name を含まないCREATE TABLEステートメントの変換が改善されました。 は、これらの一時テーブルをターゲットデータベースに保存するためのdboスキーマ AWS SCT を作成します。
Azure Synapse Analytics	Amazon Redshift	テンポラリテーブルで実行する DROP TABLE ステートメントの変換が改善されました。
Azure Synapse Analytics	Amazon Redshift	BEGIN...END ブロックを使った OBJECT_ID ステートメントの変換が改善されました。
Azure Synapse Analytics	Amazon Redshift	がブロックコメントを使用してストアドプロシージャを変換 AWS SCT できなかったエラーを解決しました。
BigQuery	Amazon Redshift	BigQuery データウェアハウスの Amazon Redshift への変換を実装しました。詳細については、「AWS SCT の BigQuery をソースとして使用する」を参照してください。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	複数のイベントを処理し、SQL Server の `inserted` および `deleted` システムテーブルを操作するトリガーの変換が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SQL Server の `inserted` および `deleted` システムテーブルで発生したリゾルバーエラーを修正しました。
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish 機能構成ファイルの新しいバージョン 2.1.0 に対するサポートを実装しました。このファイルには、特定の Babelfish バージョンでサポートされているものとサポートされていない SQL 機能が定義されています。
Oracle	Aurora MySQL MariaDB MySQL	`varchar2` データ型が誤って変換される問題を解決しました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora MySQL Aurora PostgreSQL MariaDB MySQL PostgreSQL	Oracle データベースバージョン 12c 以降では、 は次の拡張データ型 AWS SCT をサポートしています。 • VARCHAR2 • NVARCHAR2 • RAW AWS SCT これらのデータ型でサポートされる列の最大長が 8,000 バイトから 32,767 バイトに増加しました。
Oracle	Aurora PostgreSQL PostgreSQL	Oracle イベント処理パッケージで発生した解析エラーを解決しました。
Teradata	Amazon Redshift	1 つの SELECT ステートメントの複数の RESET WHEN 句に対応するアクション項目 13214 を追加しました。
Teradata	Amazon Redshift	例外処理ブロックの外にある SQLSTATE 変数用のアクション項目を追加しました。
Teradata	Amazon Redshift	ACTIVITY_COUNT 変数から ROW_COUNT への変換を実装しました。
Teradata	Amazon Redshift	組み込みジオメトリ ST_TRANSFORM 関数の変換を実装しました。
Teradata	Amazon Redshift	WHERE 句のないビューでの削除ステートメントの変換が改善されました。
Teradata	Amazon Redshift	式内の CAST 演算子の変換が改善されました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	GROUP BY 句の変換が改善されました。
Teradata	Amazon Redshift	INSTR および REGEXP_INSTR 組み込み関数の変換が改善されました。
Teradata	Amazon Redshift	横方向の列エイリアス参照が誤って変換される問題を解決しました。
Teradata	Amazon Redshift	QUALIFY サブクエリで列名が誤って変換される問題を解決しました。
Teradata	Amazon Redshift	ERRORCODE ステータス値キーワードによる .QUIT コマンドの変換を実装しました。
Teradata BTEQ	Amazon Redshift RSQL	CREATE ステートメントの変換中にアクション項目 9996 が予期せず表示される問題を解決しました。
Teradata BTEQ	Amazon Redshift RSQL	END ステートメントの変換中にアクション項目 9998 が予期せず表示される問題を解決しました。

AWS SCT ビルド 664 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	AWS SCTのデータベース移行プロジェクトのソースおよびターゲットとして、Amazon Redshift Serverless のサポートを追加しました。Amazon Redshift Serverless に接続するには、Amazon Redshift JDBC ドライバーバージョン 2.1.0.9 以降を使用していることを確認します。
すべて	すべて	変換設定ウィンドウのユーザーインターフェイスが改善されました。 AWS SCT 現在では、作成されたマッピングルールを持つデー

ソース	Target	新機能、強化点、修正点
		タベース変換ペアの設定のみが表示されます。詳細については、「マッピングルールの作成」を参照してください。
すべて	すべて	評価レポートを更新し、アクション項目の行と位置に関する重複情報を削除しました。
すべて	Amazon Redshift	データ抽出タスクに自動メモリバランシングを実装しました。
すべて	Amazon Redshift	データ抽出エージェントが AWS Snowball デバイスに接続できないというエラーを解決しました。
Azure SQL データベース IBM Db2 for z/OS IBM Db2 LUW Microsoft SQL Server MySQL Oracle PostgreSQL SAP ASE	Aurora MySQL Aurora PostgreSQL MariaDB MySQL PostgreSQL	データ抽出エージェントを実行するプラットフォームとして SUSE Linux 15.3 のサポートを実装しました。
Azure Synapse Analytics	Amazon Redshift	`DATEADD` 関数の変換が改善されました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	移行ルール内の列の照合順序を変更する機能が追加されました。
Microsoft SSIS	AWS Glue AWS Glue Studio	ユーザーがソーススクリプトを選択したときに発生する予期しないエラーを解決しました。
Oracle	Aurora MySQL MariaDB MySQL	MySQL では、ストアド関数の生成列式としての使用がサポートされていないため、ストアド関数の生成列式としての使用の変換を実装しました。 は、この動作をエミュレートするトリガー AWS SCT を作成します。
Oracle	Aurora PostgreSQL PostgreSQL	AWS SCT 拡張パックの一部として UTL_MATCH パッケージからの関数の変換を実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	NULL パラメータによる REGEXP_LIKE 関数の変換を実装しました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	SYS_EXTRACT_UTC 関数の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	Wcscats、Wcscpys、Wcsncats 関数のサポートを実装することで、C++ アプリケーションの SQL コード変換が改善されました。詳細については、「AWS SCT での C++ アプリケーションの SQL コードの変換には」を参照してください。
Oracle DW Snowflake	Amazon Redshift	変換されたステートメントに値を列データ型に明示的に変換しないという問題を解決しました。この問題は、他のテーブルからのクエリ結果を使用するステートメントで発生しました。
Teradata	Amazon Redshift	移行ルールの case sensitive と case insensitive の間で列の照合順序を変更する機能が追加されました。詳細については、「移行ルールの作成」を参照してください。
Teradata	Amazon Redshift	CREATE TABLE AS ステートメントで発生したリゾルバーエラーを修正しました。
Teradata	Amazon Redshift	COALESCE 式を含む組み込み P_INTERSECT 関数が変換されなかったエラーを修正しました。
Teradata	Amazon Redshift	Amazon Redshift で予約キーワードが使用されないように、OID という名前の列の _OID への変換を実装しました。
Teradata	Amazon Redshift	関数、プロシージャ、ビュー、マクロの RENAME ステートメントの変換を実装しました。
Teradata	Amazon Redshift	Amazon Redshift の SPLIT_PART 関数への STROKE 関数の変換を実装しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	INSTR および REGEXP_INSTR システム関数の変換が改善されました。
Teradata	Amazon Redshift	TIME データ型の変換が改善されました。
Teradata	Amazon Redshift	プライマリとセカンダリのユニークインデックスの変換を実装することで、SET および MULTISET テーブルのエミュレーションが改善されました。
Teradata	Amazon Redshift	CHARACTER 関数で発生した解析エラーを解決しました。
Teradata BTEQ	Amazon Redshift RSQL	ユーザーが Teradata Basic Teradata Query (BTEQ) スクリプトを AWS SCT プロジェクトから削除したときに発生するエラーを解決しました。

AWS SCT ビルド 663 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	移行ルールで乗算演算子を使用してchar、varchar、nvarchar、string データ型の長さを変更する機能が追加されました。詳細については、「移行ルールの作成」を参照してください。
すべて	すべて	マルチサーバー評価レポートの 3 つの新しい列のサポートを実装し、入力ファイルの形式を更新しました。入力ファイルの最新テンプレートは、必ず最新バージョンの AWS SCTで使用してください。詳細については、「データベース移行評価レポートの作成」を参照してください。
Azure Synapse Analytics	Amazon Redshift	OBJECT_ID ステートメントの変換が向上しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	データベース移行評価レポートのターゲット プラットフォームとして、Babelfish for Aurora PostgreSQL 1.2.0 の Babelfish のサポートが追加されました。詳細については、『Amazon Aurora ユーザーガイド』の「Babelfish でサポートされる機能 (バージョン別)」を参照してください。
Microsoft SQL Server DW	Amazon Redshift	`AT TIME ZONE` 句のサポートが追加されました。
Microsoft SQL Server DW	Amazon Redshift	`BEGIN/END` ブロック外のステートメントが不適切に変換される問題を解決しました。
Netezza	Amazon Redshift	`TIME` データ型の変換を改善し、関連する組み込み関数、式、リテラルの変換を実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	Oracle 10g をソースとして使用したときに発生するローダーエラーを修正しました。
Oracle	Aurora PostgreSQL PostgreSQL	`OFFSET` および `FETCH` 句の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	デフォルト値の `OUT` パラメータを含むプロシージャが誤って変換される問題を解決しました。

ソース	Target	新機能、強化点、修正点
Oracle DW	Amazon Redshift	Oracle 関数から Amazon Redshift ユーザー定義関数への変換が改善されました。
Snowflake	Amazon Redshift	WITH 句の変換が改善されました。
Teradata	Amazon Redshift	CHAR データ型でサポートされていないマルチバイト文字用の新しいアクション項目 13209 を追加しました。
Teradata	Amazon Redshift	テーブルが完全にロードされなかったローダーエラーを修正しました。
Teradata	Amazon Redshift	JOIN 条件内の組み込み P_INTERSECT 関数が変換されなかったトランスフォーマーエラーを修正しました。
Teradata	Amazon Redshift	名前に特殊文字を含むテーブルで SELECT ステートメントを実行すると、ビューの名前が大文字と小文字が間違って変換される問題を修正しました。
Teradata	Amazon Redshift	PERIOD(DATE) データ型の UNTIL_CHANGED 値を含む INSERT ステートメントの変換が改善されました。
Teradata	Amazon Redshift	Amazon Redshift の TO_CHAR 関数を使用した FORMAT 組み込み関数の変換が改善されました。
Teradata	Amazon Redshift	変換されたコードがソースコードと同じ順序で NULL 値を返すように、RANK 組み込み関数の変換が改善されました。
Teradata	Amazon Redshift	プライマリまたはセカンダリのユニークインデックスなどのユニーク制約の変換が改善されました。

AWS SCT ビルド 662 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	マルチサーバー評価レポートの作成時に、各ソースデータベースの AWS SCT プロジェクトを自動的に作成する機能が追加されました。このオプションをオンにすると、 AWS SCT はこれらのプロジェクトにマッピングルールを追加し、変換統計を保存してオフラインで使用できます。詳細については、「データベース移行評価レポートの作成」を参照してください。
すべて	すべて	マルチサーバー評価レポートを作成するときに、データベース名とスキーマ名のワイルドカードとして percent (%) がサポートされるようになりました。
すべて	Aurora MySQL Aurora PostgreSQL	すべての AWS Lambda 関数のランタイムを Python バージョン 3.9 に更新しました。
すべて	Amazon Redshift	を使用するようにすべてのデータ抽出エージェントをアップグレードしました AWS SDK for Java 2.x。
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	`NON EXISTS` 句を含む `DELETE` ステートメントの変換が改善されました。
Azure Synapse Analytics	Amazon Redshift	ソースデータベースへの接続に失敗したエラーを解決しました。

ソース	Target	新機能、強化点、修正点
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	トリガーの変換コードにオブジェクトエイリアスの 2 つの記述が含まれていたエラーを解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	「データベースオブジェクト名を大文字と小文字を区別して扱う」オプションがオンになっている場合の、大文字と小文字が混在する名前のオブジェクトの変換が改善されました。
Microsoft SQL Server DW Teradata	Amazon Redshift	PIVOT および UNPIVOT 関係演算子の変換を実装しました。
Netezza	Amazon Redshift	TIME データ型の変換を実装しました。
Oracle	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	UTL_TCP.CRLF パッケージ定数変換を実装しました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	可変長の列のデータ型の長さが変換時に維持されないという拡張パックの問題を修正しました。
Oracle	Aurora PostgreSQL PostgreSQL	C++ アプリケーションに SQL コード変換を実装しました。詳細については、「AWS SCT での C++ アプリケーションの SQL コードの変換には」を参照してください。
Oracle	Aurora PostgreSQL PostgreSQL	グローバル変数と連想配列の変換で、大文字と小文字を区別する命名のサポートを実装しました。
Oracle	Aurora PostgreSQL PostgreSQL	拡張パック内の`TO_CHAR`、`TO_DATE`、`TO_NUMBER` 関数の変換が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	`TABLE()` 演算子での変換が改善されました。
Oracle DW	Amazon Redshift	プライマリキーやその他の制約の変換に対するサポートが追加されました。

ソース	Target	新機能、強化点、修正点
Oracle DW	Amazon Redshift	条件ステートメントの変換中にアクション項目 12054 が表示されない問題を修正しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	ユーザー定義型の列を含むテーブルの変換中に、ターゲットツリーに空の名前のオブジェクトが作成されたときのエラーを解決しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	スクリプト、ルーチンなどのストアドオブジェクトのローダーエラーを修正しました。
Snowflake	Amazon Redshift	必要に応じてアクション項目 22152 が表示されず、変換結果がコメントとして AWS SCT 表示される問題を修正しました。
Snowflake	Amazon Redshift	日付と時刻関数の変換が改善され、タイムゾーンのサポートが実装されました。
Snowflake	Amazon Redshift	`WITH` 句を含む非再帰共通テーブル式 (CTE) が再帰的 CTE に変換される問題を解決しました。
Teradata	Amazon Redshift	テーブルリンクが条件になっている `UPDATE` ステートメントの変換が改善されました。
Teradata	Amazon Redshift	`RENAME TABLE` ステートメントの変換が向上しました。
Teradata	Amazon Redshift	評価レポートで、カンマ区切り値 (CSV) ファイルに空の列が表示される問題を解決しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift RSQL	変換された Basic Teradata Query (BTEQ) マクロの末尾にセミコロンがないというエラーを修正しました。
Teradata	Amazon Redshift RSQL	CASE ステートメント内の複数のデータ型値の変換が改善されました。
Teradata	Amazon Redshift RSQL	ESCAPE 文字を含む LIKE ANY 句の変換が改善されました。
Teradata	Amazon Redshift RSQL	INSERT ステートメント内の CAST 関数の変換が改善されました。
Teradata	Amazon Redshift RSQL	タイムゾーンの変換が改善され、タイムゾーンリージョンのマッピングが実装されました。
Teradata	Amazon Redshift RSQL	BTEQ スクリプトによるシェルスクリプトの変換中にアクション項目 9998 が予期せず表示される問題を解決しました。
Teradata	Amazon Redshift RSQL AWS Glue	代替変数の値に 500 文字という制限を実装しました。
Vertica	Amazon Redshift	BINARY、VARBINARY 、LONG BINARY、BYTEA、RAW データ型から VARBYTE データ型への変換を実装しました。
Vertica	Amazon Redshift	組み込み関数とリテラルの変換が改善されました。

AWS SCT ビルド 661 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	マッピングビューでマッピングルールを検索するフィルターを追加しました。フィルターを適用すると、 はサーバーマッピングリストのフィルタリング条件に一致するルール AWS SCT を表示します。詳細については、「マッピングルールの管理」を参照してください。
すべて	すべて	Apache Log4j をバージョン 2.17.1 にアップグレードしました。
すべて	Amazon Redshift	COPY コマンド内の ENCRYPTED 句を使用した Amazon Redshift へのデータ移行のサポートが追加されました。
すべて	Amazon Redshift	データ抽出エージェントの REST API を強化しました。更新された REST API では、暗号化キー、暗号化タイプなどの新しいプロパティのサポートが追加されました。
すべて	Amazon Redshift	データ抽出エージェントに役割引き受けを実装しました。今回の更新により、サブタスクの配分が改善され、 AWS SCT が指定したロールの空きエージェントにタスクを割り当てることができるようになりました。
すべて	Amazon Redshift	拡張パックを Amazon Redshift に適用する前に、必要なすべてのコンポーネントがインストールされていることを確認するようにしました。
Azure Synapse Analytics Microsoft SQL Server DW	Amazon Redshift	エラー処理のための ERROR_LINE 、ERROR_MESSAGE 、ERROR_NUMBER 、ERROR_PROCEDURE 、ERROR_SEVERITY 、ERROR_STATE システム関数の変換が改善されました。
IBM Db2 for z/OS	Aurora MySQL	AWS SCTでのデータベース移行プロジェクトのソースとして IBM Db2 for z/OS バージョン 12 のサポートが追加されました。詳細につ

ソース	Target	新機能、強化点、修正点
	Aurora PostgreSQL MySQL PostgreSQL	いては、「IBM Db2 for z/OS をソースとして使用する」を参照してください。
IBM Db2 LUW	すべて	ソースメタデータローダーを強化して、 が列名を複製するルーチンパラメータを AWS SCT ロードするようにしました。
Microsoft Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SET NOCOUNT ON 設定ステートメントを含むプロシージャのトランスフォーマーエラーを修正しました。
Microsoft Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	入力値がユーザー定義型の変数である場合の CONCAT 関数の変換が改善されました。

ソース	Target	新機能、強化点、修正点
Microsoft Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	DATEPART 関数が誤って変換される問題を解決しました。
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	新バージョンの Babelfish 機能設定ファイルのサポートを実装しました。このファイルには、特定の Babelfish バージョンでサポートされているものとサポートされていない SQL 機能が定義されています。
Microsoft SQL Server DW	Amazon Redshift	EXECUTE ステートメントを含むプロシージャが不適切に変換される問題が解決されました。
Microsoft SSIS	AWS Glue	ジョブ設定ウィザードのユーザーインターフェイスが改善されました。 は、使用可能な接続のみを接続設定セクションに表示する AWS SCT ようになりました。
Microsoft SSIS	AWS Glue	変換ルールがパッケージタスクと変数ルールに適用されない問題を解決しました。
Microsoft SSIS	AWS Glue AWS Glue Studio	サポートされていないコンポーネント用の新しいアクション項目 25042 を追加しました。
Microsoft SSIS	AWS Glue Studio	Microsoft SQL Server Integration Services (SSIS) の抽出、変換、ロード (ETL) パッケージの への変換を実装しました AWS Glue Studio。詳細については、「SSIS を AWS Glue Studio に変換する」を参照してください。
Oracle	MariaDB	MINUS 演算子の変換に関する問題を修正しました。

ソース	Target	新機能、強化点、修正点
Oracle	MariaDB	MariaDB sql_mode のシステム変数が Oracle である場合のROWNUM、SYS_GUID、TO_CHAR、ADD_MONTHS 関数の変換が改善されました。
Oracle	PostgreSQL	汎用アプリケーション変換プロジェクトでバインド変数型が SQL 型に変換されないようにするオプションが追加されました。
Oracle	PostgreSQL	汎用アプリケーション変換プロジェクトで、変換されたオブジェクトの名前にスキーマ名が追加されないようにするオプションが追加されました。
Oracle	PostgreSQL	アプリケーション SQL コード変換の ?x バインド変数形式のサポートが追加されました。
Oracle DW	Amazon Redshift	RAWデータ型から VARBYTE データ型への変換を実装しました。
Teradata	Amazon Redshift	変換されたコード内の SET テーブルをエミュレートするオプションが追加されました。このエミュレーションでは、 は MINと MAX条件 AWS SCT をサポートします。
Teradata	Amazon Redshift	異なるデータ型のパラメータを持つ結合操作の変換が改善されました。この更新により、 AWS SCT はそのようなオペレーションの変換中に変換ルールを適用できます。
Teradata	Amazon Redshift	GROUP BY 句が誤って変換される問題を解決しました。
Teradata	Amazon Redshift	QUALIFY 句が誤って変換される問題を解決しました。
Teradata	Amazon Redshift	FastExport スクリプトのインポート中に予期しないエラーが発生しました。
Teradata	Amazon Redshift RSQL	Teradata BTEQ とシェルスクリプトの変数の値を編集する機能を実装しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift RSQL	変換された Teradata FastLoad セッションのマニフェストスクリプトが欠落していた問題を解決しました。
Teradata	Amazon Redshift RSQL	変換されたFastLoad スクリプトのユニフォームリソースロケーター (URL) にマニフェストファイルの拡張子がない問題を解決しました。
Teradata BTEQ	Amazon Redshift RSQL	代替変数を含むスクリプトのローダーエラーを修正しました。
Teradata BTEQ	Amazon Redshift RSQL	アクション項目 27022 が必要なときに表示されない問題を修正しました。

AWS SCT ビルド 660 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	マルチサーバー評価レポートに AWS Secrets Manager と Secure Sockets Layer (SSL) のサポートが追加されました。詳細については、「データベース移行評価レポートの作成」を参照してください。
すべて	すべて	変換されたオブジェクトの統計収集が改善されました。
すべて	PostgreSQL	マイグレーションターゲットとして PostgreSQL メジャーバージョン 14 と MariaDB 10.6 のサポートを実装しました。
Azure Synapse Analytics	Amazon Redshift	変換されたオブジェクトの名前の変換ロジックが改善されました。

ソース	Target	新機能、強化点、修正点
Microsoft Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL	XML データ型の変換が改善されました。
Microsoft Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	NOT LIKE 句が誤って変換される問題を解決しました。
Microsoft Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	OUTPUT 句を含む INSERT、DELETE、UPDATE ステートメントでのプロシージャのトランスフォーマーエラーを修正しました。
Microsoft Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	RETURN @@ROWCOUNT ステートメントを含むプロシージャのトランスフォーマーエラーを修正しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	すべて	リンクサーバーを使用するプロシージャの変換が改善されました。
Microsoft SQL Server	すべて	マルチサーバー評価レポートに Microsoft Windows 認証のサポートを追加しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	テーブル値コンストラクターのトランスフォーマーエラーを修正しました。
Microsoft SQL Server DW	Amazon Redshift と AWS Glue	抽出、変換、ロード (ETL) スクリプトの変換が改善され、変換されたスクリプトへの正しいパスが含まれるようになりました。
Microsoft SQL Server DW	Amazon Redshift	仮想ターゲットデータベースプラットフォームと実際のターゲットデータベースプラットフォーム用に異なる変換済みスクリプトが生成される問題を解決しました。
Oracle	PostgreSQL Aurora PostgreSQL	マテリアライズドビューのインデックス変換に対するサポートが追加されました。
Oracle	PostgreSQL Aurora PostgreSQL	`NOVALIDATE` オプションで `PRIMARY KEY` コンストレイントや `UNIQUE` コンストレイントを行うと、アクション項目 5982 が表示されない問題を修正しました。

ソース	Target	新機能、強化点、修正点
Oracle DW	Amazon Redshift	変換されたスキーマに追加のカテゴリが表示される問題を解決しました。
Teradata	Amazon Redshift	未解決の列を CAST 関数の引数として変換すると、アクション項目 13185 が表示されない問題を修正しました。
Teradata	Amazon Redshift	DELETE および DELETE ALL ステートメントの変換が改善され、変換後のコードで TRUNCATE コマンドが使用されるようになりました。
Teradata	Amazon Redshift	SET テーブルの変換が改善されました。
Teradata	Amazon Redshift	NORMALIZE 条件の変換が改善されました。
Teradata	Amazon Redshift	評価レポートを更新し、データベースストレージオブジェクトのリストからデータベーススキーマ変換統計を削除しました。
Teradata	Amazon Redshift	FROM 節を含まない UPDATE ステートメントの変換が改善されました。
Teradata	Amazon Redshift	変換されたコード内の VARBYTE データ型のサポートを実装しました。
Teradata BTEQ	AWS Glue	コンテキストメニューの [AWS Glueへ変換] オプションが無効になっていた問題を解決しました。
Teradata BTEQ	Amazon Redshift RSQL	変換されたコードでデータ型が欠落していた問題を解決しました。
Teradata BTEQ	Amazon Redshift RSQL	変換されたコードで代替変数が誤って引用される問題を解決しました。

ソース	Target	新機能、強化点、修正点
Teradata BTEQ	Amazon Redshift RSQL	FastLoad スクリプト内の値で置換変数を変換する問題を修正しました。
Vertica	Amazon Redshift	変換されたコード内の TIME データ型のサポートを実装しました。
Vertica	Amazon Redshift	SELECT DISTINCT および ORDER BY 式の変換が改善されました。
Vertica	Amazon Redshift	制約の変換に対するサポートが追加されました。
Vertica	Amazon Redshift	評価レポートがカンマ区切り値 (CSV) ファイルとして保存されない問題を解決しました。

AWS SCT ビルド 659 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	複数のソースデータベースの統合評価レポートを生成する [新しいプロジェクトウィザード] を改善しました。
すべて	すべて	複数のソースデータベースとターゲットデータベースを含むプロジェクトで拡張パックが作成されなかった問題を修正しました。
すべて	すべて	アプリケーションソースコードに埋め込まれている SQL コードの変換が改善されました。
すべて	すべて	AWS SCT コマンドラインインターフェイスの異なるフォルダからスクリプトを実行する機能を追加しました。
すべて	Amazon Redshift	Amazon Redshift 仮想ターゲットデータベースプラットフォームを使用する移行プロジェクトで、ユーザーが [最適化を実行] を選択したときに表示される警告メッセージを改善しました。

ソース	Target	新機能、強化点、修正点
すべて	Aurora PostgreSQL	移行対象として、Aurora PostgreSQL 互換エディションに PostgreSQL メジャーバージョン 13 のサポートを実装しました。
すべて	Amazon RDS for MySQL	大文字と小文字を区別しないコード変換をデフォルトで実装しました。
Azure Synapse Analytics	Amazon Redshift	コマンドラインインターフェイスでソースデータベースへの接続に失敗するエラーを解決しました。
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	結合条件付きの UPDATE ステートメントを含むプロシージャの変換が改善されました。
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	等号の後の値を含むトリガー、ストアドプロシージャ、および関数の変換が改善されました。
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	DELETE ステートメントと OR 演算子を含むプロシージャのトランスフォーマーエラーを修正しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	OUTPUT 句の変換が改善されました。
Microsoft SQL Server DW	Amazon Redshift と AWS Glue	NUMERIC データ型の変換が改善されました。
Microsoft SQL Server DW	Amazon Redshift	元のテーブルと同じ名前のテーブルエイリアスを持つビューの変換が改善されました。
Microsoft SSIS	AWS Glue	AWS Glue 接続認証情報が接続の設定ウィンドウに表示されない問題を修正しました。
Netezza	Amazon Redshift	変更データキャプチャ (CDC) データ移行タスクを毎日繰り返す機能が追加されました。
Netezza	Amazon Redshift	データ抽出エージェントの登録を解除した後に [タスク] タブが非アクティブになる問題を修正しました。
Netezza	Amazon Redshift	データ移行エージェントの登録確認がユーザーインターフェイスに表示されない問題を修正しました。
Netezza	Amazon Redshift	ソースデータベースへの接続が [Loader エラー] で失敗する問題を修正しました。
Netezza	Amazon Redshift	保存したプロジェクトを開いた後にデータ移行エージェントが実行されないというエラーを解決しました。
Oracle	「Amazon RDS for Oracle」	Oracle 統合監査のサポートを実装しました。

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL Aurora PostgreSQL	C# アプリケーションに SQL コード変換を実装しました。詳細については、「C# アプリケーションでの SQL コードの変換」を参照してください。
Oracle	PostgreSQL Aurora PostgreSQL	大文字と小文字を区別するオブジェクト名の新しい変換ロジックを実装して、コード変換の変更の可視性を向上させました。 はオブジェクト名を大文字から小文字 AWS SCT に変換します。その逆も同様で、 AWS SCT は小文字のオブジェクト名を大文字に変換します。他のオブジェクト名や予約語は変更されずに変換されます。
Oracle	PostgreSQL Aurora PostgreSQL	NOT NULL 制約のないハッシュパーティションの変換が改善されました。
Oracle	Aurora PostgreSQL	Oracle CHECK、FOREIGN KEY の変換のサポートと、ENABLE NOVALIDATE 句による NOT NULL 制約が追加されました。
Oracle DW	Amazon Redshift	浮動小数点数の誤った値が移行される問題を修正しました。
Oracle DW	Amazon Redshift と AWS Glue	カンマ区切り値 (CSV) ファイルのデータベース移行評価レポートで、列が空になる問題を解決しました。

ソース	Target	新機能、強化点、修正点
SAP ASE	PostgreSQL Aurora PostgreSQL	変換が予期せず中断される問題を修正しました。
Snowflake	Amazon Redshift	VARIANT データ型の変換が改善されました。
Teradata	Amazon Redshift	COLLECT STATISTICS ステートメントの変換を向上しました。
Teradata	Amazon Redshift	PERIOD 列を含むネストされたビューを変換すると、アクション項目 9998 が表示されない問題を修正しました。
Teradata	Amazon Redshift と AWS Glue	保存したプロジェクトを開いた後、仮想 AWS Glue ターゲットプラットフォームが UI に表示されない問題を修正しました。
Teradata BTEQ	AWS Glue	保存されたプロジェクトを開いた後に仮想 AWS Glue ターゲットプラットフォームへの変換がサポートされない問題を修正しました。
Teradata BTEQ	Amazon Redshift RSQL	変換されたコードの構文強調表示が改善されました。
Teradata BTEQ	Amazon Redshift RSQL	アップロード後のパラメータ値のチェックを実装しました。サポートされていない値は [変数] タブで強調表示されます。
Vertica	Amazon Redshift	集合関数の変換を実装しました。
Vertica	Amazon Redshift	プロジェクションのマテリアライズドビューへの変換を実装し、プロジェクションのソースコードを表示する UI を改善しました。

AWS SCT ビルド 658 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	との統合を提供しました AWS Secrets Manager。Secrets Manager に保存されているデータベース接続認証情報を使用できるようになりました。
すべて	すべて	AWS SCT コマンドラインインターフェイスの YAML 形式のスクリプトのサポートが追加されました。
すべて	Amazon Redshift	データ抽出エージェントでの Amazon S3 インターフェイスエンドポイント (VPCE) のサポートを実装しました。
すべて	Amazon Redshift	既にサポートされている Amazon Redshift と AWS Glue の組み合わせに加えて、Amazon Redshift 仮想ターゲットデータベースプラットフォームのサポートが追加されました。
Greenplum	Amazon Redshift	[SQL として保存] オプションが、変換された SQL コードをファイルに保存しなかった問題を修正しました。
IBM Db2 LUW Microsoft Azure SQL データベース Microsoft SQL Server Oracle SAP ASE	Aurora MySQL	Amazon Aurora MySQL で MySQL 8.0 互換エディションの新機能をサポートするように変換が改善されました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	アクション項目 810 が必要なときに表示されない問題を修正しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	UPDATE、DELETE、INSERT ステートメントによるプロシージャの変換が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	アクション項目 7810 が必要なときに表示されない問題を修正しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	IF...ELSE ステートメント内にネストされている EXEC ステートメントの変換が改善されました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	インデックスで囲まれたビューの変換が改善されました。
Netezza	Amazon Redshift	変更データキャプチャ (CDC) 演算でロード中のライブトランザクションを追跡することで、データ移行エージェントを改善しました。CDC セッションが特定の時間に開始するようにスケジュールされている場合、データ移行タスクを停止できるようになりました。また、CDC でタスクを停止すると、コンソールにエラーログレベルが表示されます。
Oracle	すべて	が共有オプションを使用してオブジェクトを AWS SCT ロードするようにテーブルローダーを強化しました。
Oracle	Aurora PostgreSQL PostgreSQL	SYSDATE 関数の変換が改善され、[コンバージョン設定] にタイムゾーンを変更する機能が追加されました。
Oracle	Aurora PostgreSQL PostgreSQL	動的ステートメントが変換されない問題を解決しました。
Oracle	Aurora PostgreSQL PostgreSQL	変換されたコードにシステム生成の名前が含まれない問題を修正しました。

ソース	Target	新機能、強化点、修正点
Oracle Oracle DW	Aurora PostgreSQL PostgreSQL	トリガー内にネストされた SELECT ステートメントの変換が改善されました。
Oracle DW	Amazon Redshift	拡張パック内のTO_DATE、TO_TIMESTAMP 、TO_TIMESTAMP_TZ 関数の変換が改善されました。
Snowflake	Amazon Redshift	変換された SQL コードをオブジェクトごと、またはステートメントごとに異なるファイルに保存するオプションが追加されました。
Teradata	Amazon Redshift	CONCAT 関数の変換が改善されました。
Teradata	Amazon Redshift	WHERE 句内にネストされている SELECT ステートメントの変換が改善されました。
Teradata	Amazon Redshift	ユーザーがテーブルをドロップして再作成した後に、SET テーブルと MULTISET テーブルが誤って変換される問題を解決しました。
Teradata	Amazon Redshift	WITH 句を含むプロシージャの変換が改善されました。
Teradata	Amazon Redshift	DATE データ型の変換が改善されました。
Teradata	Amazon Redshift RSQL	FastExport スクリプト変換中に予期しないトランスフォーマーエラーが発生する問題を解決しました。
Teradata BTEQ	Amazon Redshift RSQL	結合インデックスからマテリアライズドビューへの変換に対するサポートが追加されました。

ソース	Target	新機能、強化点、修正点
Teradata BTEQ	Amazon Redshift RSQL	複数行を含む TITLE 定義の変換に対するサポートが追加されました。
Teradata BTEQ	Amazon Redshift RSQL	地理空間データ型のサイズが変換されない問題を解決しました。
Teradata BTEQ	Amazon Redshift RSQL	パラメータ名が小文字に変換される問題を修正しました。
Teradata BTEQ	Amazon Redshift RSQL	MACRO ステートメント内にネストされているストアドプロシージャが変換されなかった問題を修正しました。
Vertica	Amazon Redshift	ALL 演算子での変換が改善されました。
Vertica	Amazon Redshift	[変換設定]の Use Union all view? オプションが適用されなかった問題を解決しました。
Vertica	Amazon Redshift	TIME および TIME WITH TIMEZONE データ型の変換が改善されました。
Vertica	Amazon Redshift	Flex テーブルのロードに関する問題を解決しました。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 657 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	セキュリティ脆弱性の問題を解決するために、Apache Log4j をバージョン 2.17 にアップグレードしました。
すべて	Amazon Redshift	キー管理統計が AWS SCT プロジェクトに保存されないスキーマ最適化プロジェクトが改善されました。
Amazon Redshift	Amazon Redshift	サーバー情報の更新に関する問題を修正しました。
Apache Cassandra	Amazon DynamoDB	AWS SCT コマンドラインインターフェイスを使用する場合のマッピングルールの問題を修正しました。
Apache Cassandra	Amazon DynamoDB	証明書のタイトルが更新されたために移行タスクが作成されなかった問題を解決しました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	Microsoft SQL Server プロシージャをダイナミック SQL で変換しているときに、アクション項目 7672 が表示されない問題を修正しました。
Azure SQL データベース Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	テーブル値関数の変換が改善されました。
Azure SQL データベース	Aurora PostgreSQL	デフォルトの戻り値を持つストアドプロシージャの`OUT` 引数が `INOUT` 引数に変換されない問題を解決しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	PostgreSQL	
Greenplum	Amazon Redshift	最も使用頻度の高いテーブルと列を QueryLog テーブルから見つけることで、最適化戦略が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	以下の変換に関する問題が修正されました。 • 文字列連結代入演算子 (+=) • SCOPE_IDENTITY 関数 • varchar(max) データ型
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	サポートされていない関数を含むビューの変換が改善されました。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	サポートされていない関数が別の関数の引数として誤って変換される問題を修正しました。
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	遷移テーブル参照の変換が改善されました。
Microsoft SQL Server DW	Amazon Redshift	ソースデータベースのメタデータツリーに集約関数カテゴリを追加しました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server DW	Amazon Redshift	TIME データ型の変換が改善されました。
Azure Synapse Analytics Greenplum Netezza Microsoft SQL Server DW Snowflake Teradata	Amazon Redshift	仮想ターゲットデータベースプラットフォームを使用しているときに DROP および CREATE スクリプトが保存されない問題を修正しました。
Microsoft SQL Server Integration Services	AWS Glue	ソースオブジェクトのスクリプトが UI に表示されない問題を解決しました。
Netezza	Amazon Redshift	ファクトテーブルとコロケーションに適したディメンションを選択することで、最適化戦略が改善されました。
Oracle	Aurora PostgreSQL PostgreSQL	シーケンス番号を使用する Oracle トリガーが正しく変換される問題が解決されました。

ソース	Target	新機能、強化点、修正点
Oracle	Aurora PostgreSQL PostgreSQL	公開データベースリンクを含むビューの変換が改善されました。
Oracle DW	Amazon Redshift	インデックス列のカーディナリティを分析することによる最適化戦略が改善されました。
Oracle DW	Amazon Redshift	文字列を連結するユーザー定義のカスタムスカラー関数が不適切に変換される問題を修正しました。
Snowflake	Amazon Redshift	[SQL として保存] オプションが UI に表示されない問題を修正しました。
Teradata	Amazon Redshift	LOADER ERROR 例外が発生して統計収集が失敗する問題を修正しました。
Teradata	Amazon Redshift	[レポートの作成] オプションが UI に表示されない問題を修正しました。
Teradata	Amazon Redshift	CAST 関数の変換が改善されました。
Teradata	Amazon Redshift	ST_Line_Interpolate_Point の変換が破損する問題を修正しました。
Teradata	Amazon Redshift	Python ライブラリパスから予期しない値を削除しました。
Teradata	Amazon Redshift RSQL	複数の FastLoadスクリプトの変換中に表示されるリゾルバーエラーを修正しました。

ソース	Target	新機能、強化点、修正点
Teradata BTEQ	Amazon Redshift RSQL	DATABASE コマンドとジオメトリデータ型の変換が改善されました。
Teradata BTEQ	AWS Glue	UI のソーススクリプトとターゲットスクリプトが正しく同期されない問題を修正しました。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 656 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	1 つのプロジェクト内での複数のソースデータベースとターゲットデータベースのサポートが追加されました。同じプロジェクト内の異なるデータベーススキーマとターゲットプラットフォームに一致するマッピングルールを作成できるようになりました。
すべて	すべて	仮想ターゲットデータベースプラットフォームのサポートが追加されました。がソースデータベーススキーマをどのように AWS SCT 変換するかを確認するために、ユーザーはターゲットデータベースに接続する必要はありません。
すべて	すべて	UI の改良点: • ソースとターゲットのメタデータツリーに、[サーバーに接続] と [サーバーから切断] オプションが追加されました。 • AWS SCT プロジェクトからデータベースサーバーを削除するオプションを追加しました。

ソース	Target	新機能、強化点、修正点
Cassandra	Amazon DynamoDB	CASSANDRA_HOME 変数の cassandra.yaml の後ろや conf フォルダに、スラッシュ (/) が含まれない検索時の問題を解決しました。
Cassandra	Amazon DynamoDB	Amazon Linux 2 用 Amazon マシンイメージ (AMI) へのサポートが追加されました。
Cassandra	Amazon DynamoDB	Cassandra に不正なキーが与えられたときに表示されるエラーメッセージが改善されました。
Cassandra	Amazon DynamoDB	ターゲットデータベースのバージョンに応じて cassandra-env.yaml ファイル内のプロパティを変更することで、変換を改善しました。
Cassandra	Amazon DynamoDB	ターゲットの Cassandra データセンターの Java バージョンを 1.8.0 に増やしました。
Greenplum	Amazon Redshift	[Project Settings] (プロジェクト設定) での最適化戦略を改善しました。
Greenplum	Amazon Redshift	An I/O error occurred while sending to the backend エラーでオブジェクトがデータベースに適用されなかったデータ移行上の問題を解決しました。
Greenplum Microsoft SQL Server DW	Amazon Redshift	Apply RTRIM to string columns オプションが UI に表示されない問題を解決しました。
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	ターゲットプラットフォームとしての Babelfish for Aurora PostgreSQL に対するサポートを追加しました。SQL Server から Babelfish for Aurora PostgreSQL への移行を推定するための評価レポートを作成できるようになりました。
Netezza	Amazon Redshift	[Project Settings] (プロジェクト設定) での最適化戦略を改善しました。

ソース	Target	新機能、強化点、修正点
SAP ASE	Aurora PostgreSQL PostgreSQL	インデックスの一意の名前を生成する機能を実装しました。
SAP ASE	Aurora PostgreSQL PostgreSQL	ターゲットスクリプトでインデックス列が重複する問題を修正しました。
Snowflake	Amazon Redshift	[空スキーマを非表示にする]、[空データベースを非表示にする]、および [システムデータベース/スキーマを非表示にする] オプションが UI に表示されない問題を解決しました。
Teradata	Amazon Redshift RSQL	Teradata MultiLoad ジョブスクリプトを Amazon Redshift RSQL スクリプトに変換するためのサポートが追加されました。
Teradata	Amazon Redshift RSQL	および FastExport スクリプトでの FastLoad 置換変数の変換に関する問題を修正しました。
Teradata	Amazon Redshift RSQL	[概要] タブから切り替えた後、アクション項目が [アクション項目] タブに表示されない不具合を修正しました。
Teradata	Amazon Redshift RSQL	FastExport スクリプト変換中にレポートを生成した後にエラーが発生する問題を解決しました。
Teradata	Amazon Redshift RSQL	シェルスクリプトを変換した後に生じるフォーマット上の問題を解決しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift RSQL	AI 13177 が変換されたスクリプトでコメントされるように、問題を修正しました。
Teradata	Amazon Redshift	一時テーブルの変換が破損する問題を修正しました。
Teradata	Amazon Redshift	`SET QUERY_BAND` ステートメントの変換を向上しました。
Teradata	Amazon Redshift	`NORMALIZE` オペレーションが破損する問題を修正しました。
Vertica	Amazon Redshift	AI 17008 の説明を向上させました。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 655 のリリースノート

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift RSQL	FastLoad または MultiLoad が使用されている場合、すべての評価の問題がレポートに表示されるように問題を修正しました。
Teradata	Amazon Redshift RSQL	Teradata FastExport ジョブスクリプトを Amazon Redshift RSQL スクリプトに変換するためのサポートが追加されました。
Teradata	Amazon Redshift RSQL	を使用して、S3 へのマニフェストの保存アクションがオフラインモードで有効になるように問題を修正しましたFastLoad。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift RSQL	などのスクリプトにマッピングルールが適用されるように問題を修正しました FastLoad。
Greenplum	Amazon Redshift	Greenplum でサポートされる最小ドライババージョンを 42.2.5 に増加しました。
Greenplum	Amazon Redshift	ドライババージョン 42.2.5 以降で、SSL 経由での Greenplum への接続を追加しました。
Oracle DW	Amazon Redshift	別の UDF 内でカスタムユーザー定義のスカラー関数 (UDF) を実行する際のサポートが改善されました。
Oracle DW	Amazon Redshift	`Failed to compile udf` エラーで、関数がデータベースに適用されない問題を修正しました。
Oracle DW	Amazon Redshift	%ROWTYPE パラメータの `pls-type` など、適切な型宣言を行うことで変換を向上させました。
Teradata	Amazon Redshift RSQL	情報タイプ評価の問題がレポートに表示されない問題を解決しました。
Teradata	Amazon Redshift RSQL	一部のスクリプトを変換した後のトランスフォーマーエラーを解決しました。
Teradata	Amazon Redshift RSQL	問題が変換されたスクリプトにコメントされるように、問題を修正しました。
Teradata	Amazon Redshift	変換後に FastExport ->EXPORT -> 'null' が代わりに 'CAST' を表示する問題を解決しました。
Teradata	Amazon Redshift	ドライババージョン 1.2.43 を使用している場合、`Cause:[JDBC Driver]String index out of range: 0` で適用したときに拡張パックの一部の機能が失敗する問題を解決しました

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	SET テーブル変換 - SET テーブルエミュレーションを insert-select ステートメントに追加しました。
Teradata	Amazon Redshift	キャスト - サポートするデータ型キャスティングが更に追加されました。
Teradata	Amazon Redshift	"other_current_time_01" の変換の破損を修正しました
Teradata	Amazon Redshift	Teradata FastExport – Amazon Redshift RSQL: Teradata FastExport コマンドの変換を改善 — FIELD
Teradata	Amazon Redshift	Teradata FastExport – Amazon Redshift RSQL: Teradata FastExport コマンドの変換を改善 — レイアウト
Oracle	PostgreSQL Aurora PostgreSQL	SAVE EXCEPTIONS ステートメントが含まれるオブジェクトのターゲットスクリプトが再変換後に変更される問題を修正しました。
Oracle	PostgreSQL Aurora PostgreSQL	`proc_cursor_with_calc_columns` 変換後に、ORDER BY 句の誤ったフィールドが指定される問題を解決しました。
Oracle	PostgreSQL Aurora PostgreSQL	解決済み: ASSOCIATIVE COLLECTION 変換で、追加の`aws_oracle_ext$array_id$temporary` 変数宣言が必要となる。

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL Aurora PostgreSQL	解決済み：同じテーブルが所有するインデックスと同じ名前の PRIMARY KEY が誤って変換される。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 654 のリリースノート

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL Aurora PostgreSQL	階層クエリ疑似列、PRIOR 列の解析エラーに関連する問題を解決しました。
Oracle	PostgreSQL Aurora PostgreSQL	スラッシュとアスタリスク (/*) が含まれる複数行のコメントが正しく変換されるように、問題を修正しました。
Oracle	PostgreSQL Aurora PostgreSQL	拡張パックにシステムビュー USER_COL_COMMENTS エミュレーションを追加しました。

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL Aurora PostgreSQL	引用符で囲まれたリテラルの変換を改善しました。
DB2 LUW	PostgreSQL Aurora PostgreSQL	テーブル、ビュー、エイリアス、または列の説明にラベルを追加または置換する LABEL ステートメントの変換を改善しました。
Oracle	なし	SYS.USER$ システムテーブルを DBA_USERS ビューに置き換え、クエリを改善しました。
Oracle DW	Amazon Redshift	Oracle DW メタデータクエリを更新しました。
Teradata	Amazon Redshift RSQL	シェル、Teradata FastLoad、および Teradata Basic Teradata Query (BTEQ) スクリプトの Amazon Redshift RSQL スクリプトへの変換のサポートが追加されました。
Teradata BTEQ	Amazon Redshift RSQL	「merge_01」が誤って変換される問題を解決しました。
Teradata BTEQ	Amazon Redshift RSQL	新しい行のスクリプトの末尾に End または Identify (EOI) が表示される問題を修正しました。
Azure Synapse	Amazon Redshift	Azure Synapse に誤ったパスワードが与えられたときに表示されるエラーメッセージが改善されました。
Teradata	Amazon Redshift	Teradata スタンダードに従って、正しいエイリアス名を転送するように UPDATE ステートメントの変換を改善しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	アクションが受信されないカーソル変換エラーを解決しました。
Teradata	Amazon Redshift	TD_NORMALIZE_OVERLAP 変換で行が削除される問題を解決しました。
Teradata	Amazon Redshift	強化された TO_DATE 関数向けに、厳密な日付チェックがサポートされるようになりました。
Teradata	Amazon Redshift	組み込み関数 TO_NUMBER(n) の変換が改善されました。
Teradata	Amazon Redshift	メタデータツリーにスキーマカテゴリがない問題を修正しました。
Greenplum	Amazon Redshift	Greenplum テーブルの仮想パーティションを作成するときのリスト選択に、GP_SEGMENT_ID を追加しました。
Greenplum	Amazon Redshift	関数がターゲットに適用されない問題を解決しました。
MS SQL Server DW	Amazon Redshift	AI 9996 を使用せずに変換した後に、変換エラーが発生する問題を解決しました。
MS SQL Server DW	Amazon Redshift	拡張パックウィザードを開くときに、エラーがログに記録される問題を修正しました。
MS SQL Server DW	Amazon Redshift	Redshift Python 関数で誤ったスタイルのコメントが使用される問題を修正しました。
Netezza	Amazon Redshift	AWS プロファイルを持つ Netezza–Redshift 拡張パックの作成に失敗する問題を解決しました。
Teradata	Amazon Redshift RSQL	FastLoad SESSIONS コマンドの変換が改善されました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift RSQL	FastLoad スクリプト評価レポートが改善されました。
Teradata	Amazon Redshift RSQL	FastLoad WRITER Save to S3 アクションを実装しました。
Teradata	Amazon Redshift RSQL	FastLoad スクリプトの保存\s3 ボタンへのマニフェストの保存がアクティブでない問題を解決しました。
Teradata	Amazon Redshift RSQL	FastLoad multifile_script が変換後にマニフェストファイルを 1 つだけ作成し、予想される 3 つのファイルを作成する問題を解決しました。
Teradata	Amazon Redshift RSQL	S3 パスに FastLoad 余分なフォルダが表示される問題を解決しました。
Teradata	Amazon Redshift RSQL	FastLoad が S3 パスでマニフェストファイルの誤った名前を持つ問題を解決しました。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 653 のリリースノート

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL	呼び出された関数またはプロシージャで作成される動的 SQL を変換するための機能を実装しました。

ソース	Target	新機能、強化点、修正点
	Aurora PostgreSQL	
Oracle	PostgreSQL Aurora PostgreSQL	動的 SQL 変換を改善: バインド変数としてのインパラメータ。
Oracle DW 18、19	Amazon Redshift	Oracle から Redshift への変換に対する改善を実装: 強化された組み込み変換機能。集計関数 LISTAGG、分析 LISTAGG。
Oracle DW 18、19	Amazon Redshift	Oracle から Redshift への変換に対する改善を実装: 新機能のクエリ。
Vertica	Amazon Redshift	Vertica から Redshift への変換に対する改善を実装: SSL=True での SSL から JDBC への接続。
MS SQL Server DW	Amazon Redshift	MS SQL Server から Redshift への変換に対する改善:外部テーブル。
Teradata	Amazon Redshift	Teradata から Redshift への変換に対する改善: INTERVAL データ型の算術演算。
Teradata	Amazon Redshift	Teradata から Redshift への変換に対する改善: 横方向の列エイリアスをサポート。
Oracle	なし	次の Loader クエリでは、`SYS.USER$` ではなく `DBA_USERS` を使用するようになります。 • get-tree-path-list-by-name-path.sql • estimate-table-or-view-constraints-by-schema.sql • estimate-table-or-view-constraints-by-selected-schemas.sql

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	SCT が Teradata マクロを Redshift ストアドプロシージャに変換する際の、コメントの配置を改善しました。
Oracle DW	Amazon Redshift	日付/タイムスタンプ形式の要素の変換を改善: `TO_DATE`、`TO_TIMESTAMP` 、および`TO_TIMESTAMP_TZ`
Teradata	Amazon Redshift	Teradata カーソルの変換エラーを改善しました。
Teradata	Amazon Redshift	`TD_NORMALIZE_OVERLAP` の属性が変換中に削除される問題を修正しました。
Teradata	Amazon Redshift	SCT がクエリを変換するときに、`MAX` 関数は無視される問題を修正しました。
Teradata	Amazon Redshift	SCT で、テラデータの文字関数が Redshift の LENGTH 関数に変換されるようになりました。
Teradata	Amazon Redshift	SCT が、最も一般的に使用される形式の FORMAT から TO_CHAR への変換をサポートするようになりました。
すべて	すべて	暗号化されたルーチンの変換が改善されました。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 652 のリリースノート

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	PostgreSQL	`sp_getapplock` および `sp_releaseapplock` 関数向けにアプリロックを追加しました。

ソース	Target	新機能、強化点、修正点
なし	Amazon Redshift	コマンドラインインターフェイス (CLI) の改善: スクリプトコマンドモードが実装されました。
Oracle	PostgreSQL Aurora PostgreSQL	動的 SQL 内にルーチンパラメータサンプリングが実装されました。
Oracle	PostgreSQL Aurora PostgreSQL	呼び出された関数またはプロシージャで作成される動的 SQL への変換が改善されました。
Microsoft SQL Server Oracle DB2 LUW	Aurora PostgreSQL	各 Lambda 関数は、ポリシーを介して一度だけデプロイおよび構成され、一般的な Lambda 関数はすべての使用可能なソースで再利用されます。
DB2 LUW	PostgreSQL	DB2 LUW をソースとして使用しているときに「9996 — 重要度大 - トランスフォーマーエラーが発生しました」というエラーメッセージが発生する問題を解決しました。
Teradata	Amazon Redshift	近日発表の Amazon Redshift 起動時に、再帰的なテーブル表現をサポート。
Azure Synapse	Amazon Redshift	スキーマ最適化ルールを実装しました。
Teradata	Amazon Redshift	Teradata マクロから Redshift ストアドプロシージャへのタイムゾーン変換をサポート。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	PERIOD 値の算術演算をサポート。
Teradata	Amazon Redshift	Teradata 再帰共通テーブル式 (RECURSIVE CTE) の変換Support。
Teradata	Amazon Redshift	ユーザー設定の `enable_case_sensitive_identifier` を使用して、大文字と小文字を区別する識別子をサポート。このため、「COLUMN_NAME」と「Column_Name」はそれぞれ異なる列名となります。
Teradata	Amazon Redshift	10 進数フィールドが同じ精度で変換されるように、10 進数データ型上の問題を解決しました。
Teradata	Amazon Redshift	区間算術減算が正しく変換されるように、区間演算上の問題を解決しました。
Teradata	Amazon Redshift	Teradata NUMBER から DATE 型へのキャスティングを改善しました。
Teradata	Amazon Redshift	Teradata DATE から NUMBER 型へのキャスティングを改善しました。
Teradata BTEQ	Amazon Redshift	PERIOD データ型への変換が改善されました。
Teradata	Amazon Redshift	GEOMETRY 列を持つテーブルのメタデータが Teradata から正しくロードされるように、問題を修正しました。
Teradata	Amazon Redshift	Teradata マクロを Redshift ストアドプロシージャに変換するときのマージステートメントの変換をサポート。
Teradata	Amazon Redshift	Teradata から Redshift に移行する際の単純なマクロの変換を改善しました。
Teradata	Amazon Redshift	Teradata UPDATE ステートメントの変換が、Teradata 標準に従って正しいエイリアス名を転送することを確認。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 651 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	リストされている推奨変換アクション項目へのリンクを更新するための拡張 AWS SCT レポート。
MS SQL Server	PostgreSQL	STR( ) の変換に対するサポートが追加されました。
MS SQL Server	PostgreSQL	ビット単位の EXOR 演算子 (Microsoft SQL Server の ^) を PostgreSQL の # 演算子に変換する機能を追加しました。
Oracle	PostgreSQL	PostgreSQL ターゲットNULLで AWS SCT 拡張パックaws_oracle_ext.UNISTR(null)　関数が にハングする問題を解決しました。 AWS SCT 現在 が を処理していますNULL。
Teradata BTEQ	Amazon Redshift RSQL	Amazon Redshift RSQL MERGE の変換で変換エラーが発生する問題を解決するために、変換が改善されました。
Oracle DW	Amazon Redshift	拡張ビルトインを実装しました。
Oracle DW	Amazon Redshift	自動リストパーティション化 (TBL_PART_LIST_AUTO)、複数列リスト (TBL_PART_MULTI_LIST)、および間隔リファレンス (TBL_PART_RANGE_INTVAL_REF) などを含む、メタデータ機能駆動型の機能強化が追加されました。
なし	Amazon Redshift	UNION ALL 変換に使用する物理パーティションのパーティションテーブル制限を増加しました。
Teradata	Amazon Redshift	評価レポートのスコープに対する変換を改善しました。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	複雑な Teradata マクロに対する変換を改善しました。
Teradata	Amazon Redshift	サポートされていない SQL をコメントアウトしながら、Teradata マクロから Amazon Redshift ストアドプロシージャへの変換を改善しました。
Teradata	Amazon Redshift	Teradata マクロを Amazon Redshift ストアドプロシージャに変換すると、誤ったエイリアス名への参照が発生する問題を修正しました。
Teradata	Amazon Redshift	Teradata `QUALIFY` ステートメントの変換を改善しました。
Teradata	Amazon Redshift	Amazon Redshift にコメントを転送し、ビューで実行された変更の履歴を保持するための変換を改善しました。
Teradata	Amazon Redshift	RESET WHEN 句が正しく変換されない問題を解決しました。
Teradata BTEQ	Amazon Redshift	MERGE ステートメントが含まれる BTEQ スクリプトの変換が改善されました。
Teradata	Amazon Redshift	PERIOD データ型フィールドの変換を改善するための組み込み関数を追加しました。
Microsoft SQL Server	Amazon Redshift	TIME データ型の変換データ型マッピングを強化しました。
すべて	すべて	AWS Schema Conversion Tool CLI リファレンスのマニュアルの初回発行版 (PDF 形式) へのアクセスを追加しました。「AWS Schema Conversion Tool CLI リファレンス」を参照してください。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 650 のリリースノート

ソース	Target	新機能、強化点、修正点
すべて	すべて	以下を含めた抽出エージェントの使用が更新および強化されました。 • 共有ストレージと専用コピーエージェントで使用する構成。 • あるプロジェクトから別のプロジェクトへとデータを書き出すタスクのエクスポートとインポート。 • ソースとしての Azure SQL Data Warehouse (Azure Synapse) へのサポート。 • ネイティブ Netezza パーティショニングの使用。 詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。
すべて	Amazon RDS PostgreSQL 13	AWS SCT がターゲットとして Amazon RDS PostgreSQL 13 をサポートするようになりました。
Microsoft SQL Server	Aurora PostgreSQL	Microsoft SQL Server プロシージャから Aurora PostgreSQL ターゲットへの結果セットの変換が改善されました。
Oracle DW	Amazon Redshift	Oracle から Amazon Redshift への変換に対する改善を実装しました。
Oracle DW	Amazon Redshift	動的 SQL ステートメントの変換に対する改善を実装しました。
Oracle DW	Amazon Redshift	SQL UDF 変換に対する改善を実装しました。

ソース	Target	新機能、強化点、修正点
Oracle DW	Amazon Redshift	EXTERNAL TABLES の変換をサポート AWS SCT していないメッセージを明確にしました。
Oracle DW	Amazon Redshift	組み込み変換関数を強化。
Teradata BTEQ	Amazon Redshift RSQL	AWS SCT GUI の使用中に BTEQ スクリプト内の代替パラメータの処理が改善されました。
Microsoft SQL Server DW Microsoft SQL Server Azure Azure Synapse	すべて	Microsoft SQL Server、Azure、および Azure Synapse でサポートされている JDBC ドライバーの最小バージョンがアップグレードされました。

解決された問題:

- Teradata: マクロ変換に対する補足改善 [解決済み]
- SQL エラーの原因となる特殊文字がターゲットで エスケープされ、元に戻すのに再作業が必要となる問題 [解決済み]
- 全般的な機能強化

AWS SCT ビルド 649 のリリースノート

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server DW	Amazon Redshift	MSSQL から Amazon Redshift への変換が改善され、一時テーブルがサポートされるようになりました。
Oracle DW	Amazon Redshift	以下を含む、組み込み関数の拡張を実装しました。 変換関数 • TO_BINARY_DOUBLE • TO_BINARY_FLOAT • TO_NUMBER • TO_DATE • TO_TIMESTAMP • TO_TIMESTAMP_TZ • TO_DSINTERVAL • TO_YMINTERVAL • VALIDATE_CONVERSION
Oracle DW	Amazon Redshift	いかを含む、近似クエリ処理の機能強化を実装しました。 集計関数 • ANY_VALUE • APPROX_COUNT_DISTINCT •

ソース	Target	新機能、強化点、修正点
		APPROX_COUNT_DISTINCT_DETAIL • APPROX_COUNT_DISTINCT_AGG • LISTAGG • TO_APPROX_COUNT_DISTINCT
Teradata	Amazon Redshift	Teradata の自動ソートとディストリビューションキー選択に対する変換機能強化を実装しました。DB エンジンは、ディストリビューションキーとソートキーを自動的に選択します。[Current projects settings] (現在のプロジェクトの設定) > [Optimization strategies] (最適化戦略) > [Initial Key Selection Strategy] (初期キー選択戦略)] ダイアログに、[Use Amazon Redshift automatic table tuning] (Amazon Redshift の自動テーブルチューニングを使用) というラベルの付いたラジオボタンが導入されました。
Teradata	Amazon Redshift	AWS SCT が Teradata からすべてのテーブルを AWS SCT ロードするようにテーブルローダーを強化しました。
Teradata	Amazon Redshift	Amazon Redshift が単純な WHERE NOT EXISTS 句を含む相関サブクエリパターンをサポートするように、変換の強化を実装しました。
Teradata	Amazon Redshift	マクロで ECHO コマンドが使用できるように、サポートを追加しました。
DB2 LUW	PostgreSQL Aurora PostgreSQL	次を含む DYNAMIC RESULTS SETS 変換へのサポートを実装しました。 • カーソル句 WITH RETURN/WITH RETURN TO CLIENT • DYNAMIC RESULT SETS ルーチン句の変換

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server Oracle DB2 LUW SAP ASE	Aurora PostgreSQL	ターゲットとしての現在の Aurora RDS PostgreSQL へのサポートを実装しました。
Microsoft SQL Server Oracle DB2 LUW SAP ASE	MariaDB	ターゲットとしての MariaDB 10.5 へのサポートを実装しました。
Microsoft SQL Server	MariaDB	挿入された列の結果セットを返す INSERT-RETURNING へのサポートを実装しました。
Oracle	Aurora PostgreSQL	Oracle から Aurora PostgreSQL に変換するための XMLFOREST 関数へのサポートが追加されました。

解決された問題:

- 全般的な機能強化。

AWS SCT ビルド 648 のリリースノート

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL Amazon Aurora PostgreSQL 互換エディション	Aurora PostgreSQL 拡張パックカスタム適用モードを実装: 数値/日付およびテキスト型の演算子。
Oracle Microsoft SQL Server DB2 LUW	Aurora PostgreSQL	Aurora PostgreSQL Lambda Invoke 設定の実装: aws_lambda 拡張の作成、Aurora PostgreSQL クラスターへの IAM ロールの割り当て。 • Oracle - E メール、ジョブ WebAgent、キュー、ファイル • DB2 — 電子メール、タスク、ファイル • Microsoft SQL Server — 電子メール、エージェント
Oracle	PostgreSQL	FORALL ステートメント変換リファクタリングを実装: • FORALL ステートメント • FORALL ... 例外の保存 • BULK COLLECT が含まれる RETURNING INTO • SQL%BULK_EXCEPTIONS システムコレクション
Oracle DW 18、19	Amazon Redshift	Oracle から Amazon Redshift への変換に対する改善を実装: 強化された組み込み変換機能 集計関数 LISTAGG、分析 LISTAGG。

ソース	Target	新機能、強化点、修正点
Oracle DW 18、19	Amazon Redshift	Oracle から Amazon Redshift への変換に対する改善を実装: 新しい機能のクエリ。
Vertica	Amazon Redshift	Vertica から Amazon Redshift への変換に対する改善を実装: SSL=True での SSL から JDBC への接続。
Microsoft SQL Server DW	Amazon Redshift	Microsoft SQL Server から Redshift への変換に対する改善:外部テーブル。
Teradata	Amazon Redshift	Teradata から Redshift への変換に対する改善: INTERVAL データ型の算術演算。
Teradata	Amazon Redshift	Teradata から Redshift への変換に対する改善: 横方向の列エイリアスをサポート。

解決された問題:

- 全般的な機能強化

AWS SCT ビルド 647 のリリースノート

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	Microsoft SQL Server	RDS がデータベースメール機能をサポートするようになりました。
Microsoft SQL Server	MySQL	各タイプの識別子に対して最大名前を実装 — SQL Server のオブジェクト名 (テーブル、制約、列など) の最大長は 128 文字です。MySQL のオブジェクト名の最大長は 64 文字です。変換されたオブジェクトを MySQL データベースに書き込むには、名前を短くする必要があります。切り取り後の名前の重複を防ぐため、元のオブジェクト名の「チェックサム」を新しい名前に追加する必要があります。

ソース	Target	新機能、強化点、修正点

		64 文字を超える名前は次のように切り取られます。
		
		`[first N chars]() + "" + [checksum]()`
		
		`[first N chars] = 64 - 1 - [length of checksum string]`
		
		例:
		`example_of_a_test_schema_with_a_name_length_greater_than_64_characters ?? example_of_a_test_schema_with_a_name_length_greater_than_64_9703`

Oracle	MySQL/Aurora MySQL	ストレージオブジェクトに対するコメントのロードと変換を実装しました。例えば、テーブルに対するコメントの処理や、テーブル/ビュー列に対するコメントの処理などです。
Teradata	Amazon Redshift	TIME データ型変換へのサポートが追加されました。
Teradata	Amazon Redshift	変換を改善 — TD_NORMALIZE_OVERLAP を実装しました。
Microsoft SQL Server DW	Amazon Redshift	変換を改善 — WITH 句が含まれる SELECT、FROM が含まれない SELECT
すべて	すべて	AWS SCT Data Migration Service Assessor (DMSA) — この新しい機能により、複数のサーバーを評価し、環境に最適なターゲットの方向性を示す概要レポートを受け取ることができます。
すべて	すべて	AWS SCT ウィザード — ターゲット比較では、単一のテーブルビューにターゲット間の違いが表示されるようになりました。
すべて	すべて	ツリーフィルター UI — 改良されたメタデータフィルターが、より複雑なフィルタリングパターンを処理します。

ソース	Target	新機能、強化点、修正点
すべて	すべて	評価レポート — 改良された「警告」セクションでは、問題の説明を向上し、より明確に理解できるようにしました。

解決された問題:

- 全般的な機能強化
- データエクストラクタ — サブタスクが ConcurrentModificationException 〔解決済み] で失敗しました。
- Microsoft SQL Server から MySQL へ — 識別子の最大長 [解決済み]。

AWS SCT ビルド 646 のリリースノート

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL	TM フォーマットモデルの実装が改善されました。
Oracle	PostgreSQL	SP 形式のマスク実装により、英語のみ SP サフィックスへの基本サポートが提供されます。
Oracle	PostgreSQL	Oracle long object names handling — は、ターゲットの最大識別子長属性に従って Oracle long object names を処理する AWS SCT ようになりました。
	Amazon Redshift	を使用した Amazon Redshift エンコーディング AZ64 AWS SCT — 一部のデータ型に圧縮エンコーディング AZ64 を追加
Teradata	Amazon Redshift	暗示的トランザクション変換へのサポートが追加されました。
Teradata	Amazon Redshift	Teradata 地理空間組み込み関数 `ST_LineString` メソッドへのサポートが追加されました。

ソース	Target	新機能、強化点、修正点
Greenplum	Amazon Redshift	Greenplum シーケンス変換 — [Properties] (プロパティ) タブに次の項目を追加:[min value] (最小値)、[max value] (最大値)、[increment] (増分)、[cycle] (サイクル)。
Greenplum	Amazon Redshift	リゾルバ —「char」データ型の解決を追加しました。
Greenplum	Amazon Redshift	文字変換の長さ — 文字タイプの PL/pgSQL 変換を更新しました。
Greenplum	Amazon Redshift	Greenplum ディストリビューションキーの選択で、テーブルに分散キーがあっても、テーブル AWS SCT をランダム分散として認識してフェッチできなかった問題を解決しました。
Teradata	Amazon Redshift	Teradata カーソルのサポート — カーソル変換に対するサポートが追加されました。
Teradata	Amazon Redshift	アイデンティティ列 — アイデンティティ列変換に対するサポートが追加されました。
Teradata	Amazon Redshift	INTERVAL データ型 — INTERVAL データ型の変換に対するサポートが追加されました。

解決された問題:

- 全般的な機能強化
- Greenplum: ログのエラーのため、変換を実行できません [解決済み]。
- MSSQL — PostgreSQL: LAG 関数の変換時にトランスフォーマーエラーが発生 [解決済み]。
- MSSQL — PostgreSQL: SCOPE_IDENTITY [解決済み]。
- AWS SCT DW プロジェクトでハングイン [解決済み〕。
- AWS SCT 〔解決済み] の列名に追加のスペースを削除するためのマッピングルールが必要です。

AWS SCT ビルド 645 のリリースノート

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	Teradata の非完全修飾ビュー (ビュー名またはビュー内の非完全修飾オブジェクト) を解決するためのソリューションを提供。
Teradata	Amazon Redshift	ノードのコンピューティングのための ASCII 関数へのサポートが追加されました。
Teradata	Amazon Redshift	が としてCHAR定義された Teradata 内のマルチバイトデータ AWS SCT を検出するとCHAR(N)、Amazon Redshift VARCHAR(3*N) でに変換されます。
Teradata	Amazon Redshift	日付と数値の間での Teradata CAST を提供します。 • SELECT Cast('2020-07-17' AS BIGINT) • SELECT Cast(20200630 - 19000000 AS DATE)
Teradata	Amazon Redshift	Teradata PERIOD データ型の 2 つのAmazon Redshift TIMESTAMP 列への変換をサポート。 • PERIOD(TIMESTAMP) • PERIOD(TIMESTAMP WITH TIMEZONE)
Teradata	Amazon Redshift	RESET WHEN 句が含まれる Teradata RANK 関数の変換をサポート。
Teradata	Amazon Redshift	明示的なデータ型変換における CAST 、および表現の暗黙的な CAST へのサポートを改善しました。
Teradata	Amazon Redshift	サポートされていない相関サブクエリパターンを報告します。詳細については、Amazon Redshift データベースデベロッパーガイドの「相関性のあるサブクエリ」を参照してください。

ソース	Target	新機能、強化点、修正点
なし	Amazon Redshift	RA3 ノードタイプのテーブル制限に対するサポートを改善。
Teradata	Amazon Redshift	Teradata 地理空間データ抽出に対するサポートが追加されました。詳細については、『Amazon Redshift データベースデベロッパーガイド』の「Amazon Redshift での空間データのクエリ」を参照してください。
Microsoft SQL Server	PostgreSQL	オプション `convert_procedures_to_function` が追加されました。

解決された問題:

- 全般的な機能強化

AWS SCT ビルド 644 のリリースノート

AWS SCT リリース 1.0.643 の変更は AWS SCT 1.0.644 リリースにマージされます。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	複数の変換機能を改善。 • テーブルエイリアスによる `QUALIFY` を使用した変換を改善。 • `IN` 演算子での変換を改善。 • `LIKE` 演算子での変換を改善。 • 変換されたコード内の問題点をハイライト表示することで、変換を改善。 • SQL の `WHERE`、`QUALIFY` 句の異常な順序での変換を改善。

ソース	Target	新機能、強化点、修正点
		• UPD_FT_SVC_TRANS_BH_CBH_IND プロシージャの JOIN() 構築変換中に発生するトランスフォーマーエラーを修正しました。 • マクロからストアドプロシージャへの変換が改善されました。 提供された sql/bteq スクリプトを解析し、ソースコードで発生した構文構造の数に関するレポートを生成できる特別な AWS SCT CLI コマンドを追加しました。 • BTEQ コマンドの数 • ハンドラの数 • CAST ケースの数 • DML/DDL ケースの数 • 更新可能なビュー上の DML の数 評価レポートのアクション項目を追加: カスタム日付形式の Teradata 列は、Amazon Redshift ではサポートされていません。
Oracle	PostgreSQL/Aurora PostgreSQL	拡張パックインストールスクリプトを保存する機能を追加しました。 AI 5334 の重要度レベルを変更しました。 レコードをパッケージ変数 IMPLEMENTATION として使用する際のパフォーマンスが向上しました。 XMLAGG 集計関数サポートを追加

ソース	Target	新機能、強化点、修正点
IBM Db2	PostgreSQL/Aurora PostgreSQL	ストレージオブジェクトの実装におけるコメントのロードと変換を追加。
MS SQL DW	Amazon Redshift	変換を改善: PATINDEX の問題を解決しました。 UI の改良点: • ソースツリー実装にSQL として保存 • 複数のファイルのスクリプト生成ロジックを追加。
Vertica	Amazon Redshift	UI の改善: ソースツリー実装にSQL として保存。

解決された問題:

- Teradata と Amazon Redshift 間の変換に対する全般的な機能強化
- 一般的なバグ修正と UI 向上

AWS SCT ビルド 642 のリリースノート

AWS Schema Conversion Tool リリース 1.0.642 の変更点。

> **Note**
>
> AWS Schema Conversion Tool （ AWS SCT) ビルド 1.0.642 の変更は、Windows、Ubuntu、および Fedora に適用されます。macOS 向けの 1.0.642 ビルドはありません。

ソース	Target	新機能、強化点、修正点
Microsoft SSIS	AWS Glue	Microsoft SQL Server Integration Services (SSIS) ETL パッケージのへの変換を実装しました AWS Glue。詳細については、「AWS SCT を使用して SSIS をAWS Glue に変換する」を参照してください。
Oracle	MariaDB/SQL MODE=CLE/MySQL/Amazon Aurora MySQL	WITH 句に PL/SQL 宣言セクションを実装しました。
Oracle	PostgreSQL/Aurora PostgreSQL	[DBMS_SESSION.RESET_PACKAGE] と [DBMS_SESSION.MODIFY_PACKAGE] のサポートが追加されました。
Vertica	Amazon Redshift	Vertica データベースから Amazon Redshift への SQL スクリプトのエクスポートが有効になりました。

解決された問題:

- 評価レポートの強化
- 評価レポート UI の強化
- UI から JVM 設定を変更する機能の追加
- 全般的な機能強化。

AWS SCT ビルド 641 のリリースノート

AWS Schema Conversion Tool リリース 1.0.641 の変更点。

> **Note**
>
> AWS Schema Conversion Tool （ AWS SCT) ビルド 1.0.641 の変更は、Windows、Ubuntu、および Fedora に適用されます。macOS 向けの 1.0.641 ビルドはありません。

ソース	Target	新機能、強化点、修正点
Oracle/ MS SQL/ MySQL/ PostgreSQL/Db2 LUW	すべて	.csv ファイルで時間レポート計算を作成。
Teradata	Amazon Redshift	CSUM 関数に対するサポートが追加されました。 Teradata 地理空間データ型に対するサポートが追加されました。
Teradata	すべて	アイデンティティ列の変換に対するサポートが追加されました。
Greenplum	Amazon Redshift	Greenplum テーブル変換中のディストリビューションスタイル AUTO に対するサポートが追加されました。
SAP ASE	すべて	.csv ファイルで時間レポート計算を作成。

解決済み:

- さまざまなバグを修正しました。
- 様々なパフォーマンスが改善されました。

AWS SCT ビルド 640 のリリースノート

AWS SCT リリース 1.0.633、1.0.634、1.0.635、1.0.636、1.0.637、1.0.638、1.0.639、および 1.0.640 の変更は、 AWS SCT 1.0.640 リリースにマージされます。

> **Note**
>
> AWS SCT ビルド 1.0.640 の変更は、Windows、Ubuntu、および Fedora に適用されます。MacOS には適用されません。
> Apple macOS に AWS SCT バージョン 1.0.640 以降をインストールすることはできません。AWS SCT バージョン 1.0.632 は、Apple macOS でのインストールをサポートする最後のバージョンでした。

次の表に、リリース 1.0.640 に結合された AWS Schema Conversion Tool バージョンの機能とバグ修正の一覧を示します。これらのテーブルは、ソースエンジンによる機能とバグ修正をグループ化します。

トピック

リリース 1.0.640 Oracle の変更点

次の表に、Oracle がソースエンジンであるビルド 1.0.640 の変更を示します。

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL Aurora PostgreSQL	Java アプリケーションと Pro*C アプリケーションに SQL コード変換を実装しました。
Oracle	PostgreSQL Aurora PostgreSQL	WHERE 句で使用する場合、次の関数のパフォーマンスが向上しました。 • aws_oracle_ext.to_date • aws_oracle_ext.to_char • aws_oracle_ext.to_number • aws_oracle_ext.sysdate • aws_oracle_ext.sys_context

ソース	Target	新機能、強化点、修正点
Oracle	RDS MariaDB 10.4	すべてのオンライントランザクション処理 (OLTP) ベンダーに対して RDS MariaDB 10.4 のサポートが追加されました。
Oracle	PostgreSQL/Aurora PostgreSQL	DBMS_UTILITY.GET_TIME のサポートが追加されました。 以下のエミュレーションが追加されました。 • DBMS_UTILITY.GET_TIME • DBMS_UTILITY.FORMAT_CALL_STACK • DBMS_UTILITY.CURRENT_INSTANCE
Oracle	MariaDB/MySQL/Aurora MySQL/Microsoft SQL Server Mode=Oracle/PostgreSQL/Aurora PostgreSQL/RDS Oracle	TABLE(DATA,EXTENDED DATA)、VIEW(DATA,EXTENDED DATA)、および SEQUENCE(DATA) の共有句のサポートを追加
Oracle	PostgreSQL/Aurora PostgreSQL/Oracle RDS	カラムの DEFAULT 定義を拡張して、明示的な NULL 挿入に DEFAULT を適用できます。 DEFAULT 句には、新しい ON NULL 句があります。この新しい句は、INSERT ステートメントが NULL と評価される値を割り当てようとしたときに、指定されたデフォルトのカラム値を割り当てるようにデータベースに指示します。

ソース	Target	新機能、強化点、修正点
Oracle	MariaDB/MariaDB (SQL MODE=ORACLE)	挿入時に自動的にインクリメントする「Identity Columns」のサポートが追加されました。
すべて	すべて	Amazon Corretto JDK 8 から 11 にアップグレードします。ダウンロードリンクなどの詳細については、「Amazon Corretto 11 ユーザーガイド」の「Amazon Corretto 11 とは」を参照してください。
すべて	すべて	評価レポートに、ユーザーのデータベース内の不整合の可能性についての情報を追加しました。
Oracle	MariaDB 10.2/MariaDB 10.3/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	`DEFAULT` 句には新しい `ON NULL` 句があり、INSERT 文が `NULL` に評価される値を割り当てようとしたときに、指定されたデフォルトのカラム値を割り当てるようにデータベースに指示します。
Oracle	Oracle RDS/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	`IDENTITY` のサポートが追加されました。

ソース	Target	新機能、強化点、修正点
Oracle	MySQL 8.x	CHECK 制約のサポートが追加されました。
Oracle	PostgreSQL/Aurora PostgreSQL	拡張パックルーチンを使用して、ANYDATA IS NULL/IS NOT NULL のチェックを実装しました。 XMLSequence の TABLE 関数を基に、クエリで使用する VALUE 関数のエミュレーションを実装しました。 次の組み込みルーチンに対する DBMS_LOB サポートが追加されました。 • DBMS_LOB.CREATETEMPORARY • DBMS_LOB.FREETEMPORARY • DBMS_LOB.APPEND
すべて	SQL Server	SQL Server 2019: 新しいインデックス属性 ANYDATA IS NULL/IS NOT NULL のサポートが追加されました。 SQL Server 2017: グラフデータベースノードとエッジテーブルタイプのサポートが追加されました。 SQL Server 2016: TEMPORAL TABLES のサポートが追加されました。
すべて	すべて	物理パーティションを仮想パーティションで上書きする機能を実装しました。データウェアハウスの抽出は、作成された仮想パーティションに従ってデータを抽出します。

ソース	Target	新機能、強化点、修正点
Oracle	Amazon Redshift	ネストされたブロック内のカーソル属性の変換を実装しました。 Amazon Redshift はコレクションをサポートしていません。関連する変数は VARCHAR として変換されます。変数を別の変数に割り当てる以外のすべてのコレクションオペレーションは、開始および コレクション要素のアクセスを含め、拒否されます。 Amazon Redshift の分散スタイル = AUTO を実装しました。

ソース	Target	新機能、強化点、修正点
Oracle	PostgreSQL/Aurora PostgreSQL	Oracle の予約されていない単語が PostgreSQL で予約されている場合、次のことが当てはまります。 • 単語が引用符で囲まれている場合、大文字小文字を保持し、引用符で囲まれます。 • 単語が引用符で囲まれていない場合は、大文字にキャストされ、引用符で囲まれます。 LTRIM、RTRIM、TRIM 関数への入力機能を実装しました。 SELECT DISTINCT、ORDER BY 式は選択リストに表示する必要があります。 DEFAULT 値を持つパラメータの後に続くカーソルパラメータの場合、DEFAULT IS NULL AWS SCT 句を追加します。 Source OUT カーソルパラメータは、IN カーソルパラメータに変換されます。 [Conversion settings] (変換設定) に [Package variables logic implementation] (パッケージ変数ロジックの実装) オプションを追加し、パッケージ変数を再実装しました。使用可能な設定は、[session variables] と [plv8 global objects] です。デフォルトは [session variables] です。 dblink と pg_background による AUTONOMOUS_TRANSACTION サポートを実装しました。
Oracle	すべて	SYS_%_TAB_COMMENTS の表示を実装しました。
Oracle	PostgreSQL	PostgreSQL では、フィルタへの変数入力はサポートされていません。Oracle から PostgreSQL への変換時に、変数フィルタで例外が発生すると報告されるようになりました。

ソース	Target	新機能、強化点、修正点
Oracle	Amazon Redshift	FOR..LOOP カーソル変換の改善されたストアドコードを実装しました。 デフォルトのパラメータを持つ関数/プロシージャのストアドコード呼び出しを実装しました。 WHERE 句なしのエイリアスで UPDATE するストアドコード機能を実装しました。 SELECT FROM dual で追加のケースを実行するストアドコード関数を実装しました。 ストアドコードテーブル Table%ROWTYPE パラメータとパッケージ変数を実装しました。 JAVA と外部プロシージャで使用するストアドコードを実装しました。 ストアドコードに標準 Oracle パッケージを実装しました。

リリース 1.0.640 Microsoft SQL サーバーの変更点

次の表は、Microsoft SQL Server がソースエンジンであるビルド 1.0.640 の変更を示しています。

ソース	Target	新機能、強化点、修正点
Microsoft Azure/ Microsoft SQL Server	PostgreSQL/Aurora PostgreSQL/MySQL/ Aurora MySQL	COLUMN STORE インデックスのサポートが追加されました。

ソース	Target	新機能、強化点、修正点
Microsoft SQL Server	RDS MariaDB 10.4	すべてのオンライントランザクション処理 (OLTP) ベンダーに対して RDS MariaDB 10.4 のサポートが追加されました。
Azure/SQL Server	MariaDB/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	OPTIMIZE_FOR_SEQUENTAL_KEY インデックス属性のサポートが追加されました。
Azure/SQL Server	MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	データベースノードおよびエッジテーブルタイプのサポートが追加されました。
Azure/SQL Server	MariaDB/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	TEMPORAL TABLES のサポートが追加されました。

ソース	Target	新機能、強化点、修正点
すべて	すべて	Amazon Corretto JDK 8 から 11 にアップグレードします。ダウンロードリンクなどの詳細については、「Amazon Corretto 11 ユーザーガイド」の「Amazon Corretto 11 とは」を参照してください。
すべて	すべて	評価レポートに、ユーザーのデータベース内の不整合の可能性についての情報を追加しました。
Azure/SQL Server	MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL/MariaDB	SQL Server Graph Architecture の DML 処理のサポートが追加されました。
SQL Server	Aurora PostgreSQL	`par_` プレフィックスなしでパラメータを変換するオプションが追加されました。
Azure/SQL Server	MySQL 8.x	CHECK 制約のサポートが追加されました。
すべて	SQL Server	SQL Server 2019: 新しいインデックス属性 ANYDATA IS NULL/IS NOT NULL のサポートが追加されました。 SQL Server 2017: グラフデータベースノードとエッジテーブルタイプのサポートが追加されました。 SQL Server 2016: TEMPORAL TABLES のサポートが追加されました。
すべて	すべて	物理パーティションを仮想パーティションで上書きする機能を実装しました。データウェアハウスの抽出は、作成された仮想パーティションに従ってデータを抽出します。

ソース	Target	新機能、強化点、修正点
SQL Server	AWS Glue (Python シェル)	以下のような変換の改善。 • Python.String への組み込み関数の変換を実装しました。 • ストアドコードに EXECUTE と EXEC を実装しました。 • テーブルタイプを使用して実装されます。
Azure/SQL Server	PostgreSQL/Aurora PostgreSQL	オプションで、$TMP プロシージャを実装しました。
SQL Server	MySQL/ Aurora MySQL	日付を使用する拡張された算術演算。 コンストラクションエミュレーション 'TOP (式) WITH TIES。 生成された refcursor を出してプロシージャを呼び出した後、refcursor が閉じるようになりました。 GLOBAL 独立性レベルの設定は、Aurora MySQL ではサポートされていません。セッションスコープのみを変更できます。トランザクションのデフォルトの動作では、REPEATABLE READ と整合性のある読み込みが使用されます。READ COMMITTED で使用するように設計されたアプリケーションは、変更が必要な場合があります。または、デフォルトを READ COMMITTED に明示的に変更することもできます。

ソース	Target	新機能、強化点、修正点
SQL Server	AWS Glue (Python シェル)	SQL Server ステートメントは完全な結果セットを生成しますが、1 行ずつ処理すると最もよい結果になる場合があります。結果セット上でカーソルを開くと、一度に 1 行ずつ結果セットを処理できます。カーソルデータ型の変数またはパラメータにカーソルを割り当てることができます。 ストアドコードの一連の Transact-SQL ステートメントを囲むように実装し、Python が SQL Server の BEGIN と END をフロー制御としてサポートしていない場合でも、Transact-SQL ステートメントのグループを実行できるようにしました。 SQL Server LABEL ステートメントと GOTO ステートメントは、 ではサポートされていません AWS Glue。 AWS SCT がコード内でラベルを検出した場合、そのラベルはスキップされます。 AWS SCT が GOTO ステートメントを検出すると、コメントされます。
SQL Server	Amazon Redshift	IF ...ELSE control を実装することで、ストアドコードの Transact-SQL ステートメントの条件付き処理を 実装しました。 Transact-SQL ステートメントのグループをブロックとして実行できるように、ストアドコードの一連の Transact-SQL ステートメントを囲むように実装しました。ネストされた BEGIN ...END blocks をサポート ... END ブロック。 ストアドコードに SET と SELECT を実装しました。 テーブルにユーザー指定のソートキーを作成することで、CREATE INDEX を Amazon Redshift (インデックスをサポートしていません) に実装しました。

リリース 1.0.640 MySQL の変更点

次の表に、MySQL がソースエンジンであるビルド 1.0.640 の変更を示します。

ソース	Target	新機能、強化点、修正点
MySQL	PostgreSQL 12.x	生成されたカラムのサポートが追加されました。
すべて	すべて	Amazon Corretto JDK 8 から 11 にアップグレードします。ダウンロードリンクなどの詳細については、「Amazon Corretto 11 ユーザーガイド」の「Amazon Corretto 11 とは」を参照してください。
すべて	すべて	評価レポートに、ユーザーのデータベース内の不整合の可能性についての情報を追加しました。
MySQL	PostgreSQL/Aurora PostgreSQL 11。	以下のサポートを追加しました。 • SQL ストアドプロシージャ内の埋め込みトランザクション。 • SQL ストアドプロシージャを CALL する機能。 • SQL ストアドプロシージャを作成する機能。
すべて	SQL Server	SQL Server 2019: 新しいインデックス属性 ANYDATA IS NULL/IS NOT NULL のサポートが追加されました。 SQL Server 2017: グラフデータベースノードとエッジテーブルタイプのサポートが追加されました。 SQL Server 2016: TEMPORAL TABLES のサポートが追加されました。
すべて	すべて	物理パーティションを仮想パーティションで上書きする機能を実装しました。データウェアハウスの抽出は、作成された仮想パーティションに従ってデータを抽出します。

リリース 1.0.640 PostgreSQL の変更点

次の表は、PostgreSQL がソースエンジンであるビルド 1.0.640 の変更を示しています。

ソース	Target	新機能、強化点、修正点
PostgreSQL	MySQL 8.x	MySQL は、列値ではなく式値にインデックスを付ける機能的なインデックスキー部分の作成をサポートするようになりました。機能的なキーの部分によって、JSON 値のようなインデックスを作成できない値のインデックスを作成できるようになります。 MySQL では、Now CTE および Recursive CTE がサポートされるようになりました。
すべて	すべて	Amazon Corretto JDK 8 から 11 にアップグレードします。ダウンロードリンクなどの詳細については、「Amazon Corretto 11 ユーザーガイド」の「Amazon Corretto 11 とは」を参照してください。
すべて	すべて	評価レポートに、ユーザーのデータベース内の不整合の可能性についての情報を追加しました。
PostgreSQL 11.x	PostgreSQL/Aurora PostgreSQL 11。	以下のサポートを追加しました。 • SQL ストアドプロシージャ内の埋め込みトランザクション。 • SQL ストアドプロシージャを CALL する機能。 • SQL ストアドプロシージャを作成する機能。
PostgreSQL	MySQL 8.x	インデックスの降順に対する MySQL のサポートが追加されました。インデックス定義内の DESC は無視されませんが、キー値は降順で保存されます。 MySQL では、BLOB、TEXT、GEOMETRY、JSON データ型のデフォルト値としての式を含む、データ型仕様におけるデフォルト値として式の使用のサポートが追加されました。 いくつかの既存の集計関数がウィンドウ関数として使用できるようになりました。 • `AVG()` • `BIT_AND()`

ソース	Target	新機能、強化点、修正点
		• BIT_OR() • BIT_XOR() • COUNT() • JSON_ARRAYAGG() • JSON_OBJECTAGG() • MAX() • MIN() • STDDEV_POP() • STDDEV() • STD() • STDDEV_SAMP() • SUM() • VAR_POP() • VARIANCE() • VAR_SAMP() MySQL は、クエリからの各行に対して、その行に関連する行を使用して計算を実行するウィンドウ関数をサポートしています。 • CUME_DIST() • DENSE_RANK() • FIRST_VALUE() • LAG() • LAST_VALUE() • LEAD() • NTH_VALUE() • NTILE() • PERCENT_RANK() • RANK()

ソース	Target	新機能、強化点、修正点
		• ROW_NUMBER()
PostgreSQL	MySQL 8.x	CHECK 制約のサポートが追加されました。
すべて	SQL Server	SQL Server 2019: 新しいインデックス属性 ANYDATA IS NULL/IS NOT NULL のサポートが追加されました。 SQL Server 2017: グラフデータベースノードとエッジテーブルタイプのサポートが追加されました。 SQL Server 2016: TEMPORAL TABLES のサポートが追加されました。
すべて	すべて	物理パーティションを仮想パーティションで上書きする機能を実装しました。データウェアハウスの抽出は、作成された仮想パーティションに従ってデータを抽出します。
PostgreSQL/Aurora PostgreSQL	すべて	システムビュー sysindexes エミュレーションが追加されました。 INTO を指定せずにプロシージャに SELECT 文がある場合、ターゲット上のプロシージャに対して refcursor 型のパラメータ INOUT p_refcur が作成されます。

リリース 1.0.640 Db2 LUW の変更点

次の表は、DB2 LUW がソースエンジンであるビルド 1.0.640 の変更を示しています。

ソース	Target	新機能、強化点、修正点
DB2 LUW	RDS MariaDB 10.4	すべてのオンライントランザクション処理 (OLTP) ベンダーに対して RDS MariaDB 10.4 のサポートが追加されました。

ソース	Target	新機能、強化点、修正点
すべて	すべて	Amazon Corretto JDK 8 から 11 にアップグレードします。ダウンロードリンクなどの詳細については、「Amazon Corretto 11 ユーザーガイド」の「Amazon Corretto 11 とは」を参照してください。
すべて	すべて	評価レポートに、ユーザーのデータベース内の不整合の可能性についての情報を追加しました。
DB2 LUW	MySQL 8.0.17	CHECK 制約のサポートが追加されました。
すべて	SQL Server	SQL Server 2019: 新しいインデックス属性 ANYDATA IS NULL/IS NOT NULL のサポートが追加されました。 SQL Server 2017: グラフデータベースノードとエッジテーブルタイプのサポートが追加されました。 SQL Server 2016: TEMPORAL TABLES のサポートが追加されました。
すべて	すべて	物理パーティションを仮想パーティションで上書きする機能を実装しました。データウェアハウスの抽出は、作成された仮想パーティションに従ってデータを抽出します。

リリース 1.0.640 Teradata の変更点

次の表は、Teradata ソースエンジンのビルド 1.0.640 の変更を示しています。

ソース	Target	新機能、強化点、修正点
Teradata	Amazon Redshift	MERGE ステートメントと QUERIFY ステートメントのサポートが追加されました。 Teradata ステートメントからLOCKING ROWS FOR ACCESS 句を削除しました。

ソース	Target	新機能、強化点、修正点
		CAST 関数のサポートが追加されました。
すべて	すべて	Amazon Corretto JDK 8 から 11 にアップグレードします。ダウンロードリンクなどの詳細については、「Amazon Corretto 11 ユーザーガイド」の「Amazon Corretto 11 とは」を参照してください。
Teradata	Teradata	REGEXP_INSTR () と REGEXP_SUBSTR () の改良を実装しました。
すべて	すべて	評価レポートに、ユーザーのデータベース内の不整合の可能性についての情報を追加しました。
すべて	SQL Server	SQL Server 2019: 新しいインデックス属性 ANYDATA IS NULL/IS NOT NULL のサポートが追加されました。 SQL Server 2017: グラフデータベースノードとエッジテーブルタイプのサポートが追加されました。 SQL Server 2016: TEMPORAL TABLES のサポートが追加されました。
Teradata	すべて	REGEXP_INSTR() と REGEXP_SUBSTR() のサポートが追加されました。
すべて	すべて	物理パーティションを仮想パーティションで上書きする機能を実装しました。データウェアハウスの抽出は、作成された仮想パーティションに従ってデータを抽出します。
Teradata	Amazon Redshift	プロジェクト設定、SQL として保存および適用、ドロップダウンリスト: 単一ファイル/複数ファイルの設定を使用して、ソースツリーの SQL をステージごとに単一ファイルまたは複数ファイルに保存する機能を実装しました。 ビューとプロシージャの変換の改善を実装しました。
Teradata	すべて	Teradata バージョン 16.20 のサポートが追加されました。

他のエンジンに対するリリース 1.0.640 の変更点

次の表に、他のソースエンジンに対するビルド 1.0.640 の変更を示します。

ソース	Target	新機能、強化点、修正点
Sybase	RDS MariaDB 10.4	すべてのオンライントランザクション処理 (OLTP) ベンダーに対して RDS MariaDB 10.4 のサポートが追加されました。
SAP ASE	MariaDB	以下を実装しました。 • MariaDB 10.4 • EXECUTE IMMEDIATE ステートメント • DEFAULT 定義 • CHECK 制約のサポート
SAP ASE	PostgreSQL 12.x	生成されたカラムのサポートが追加されました。
すべて	すべて	Amazon Corretto JDK 8 から 11 にアップグレードします。ダウンロードリンクなどの詳細については、「Amazon Corretto 11 ユーザーガイド」の「Amazon Corretto 11 とは」を参照してください。
すべて	すべて	評価レポートに、ユーザーのデータベース内の不整合の可能性についての情報を追加しました。
SAP ASE	MySQL 8.0.17	CHECK 制約のサポートが追加されました。
すべて	SQL Server	SQL Server 2019: 新しいインデックス属性 ANYDATA IS NULL/IS NOT NULL のサポートが追加されました。 SQL Server 2017: グラフデータベースノードとエッジテーブルタイプのサポートが追加されました。

ソース	Target	新機能、強化点、修正点
		SQL Server 2016: TEMPORAL TABLES のサポートが追加されました。
Vertica	Amazon Redshift	分散スタイル = AUTO のサポートが追加されました。
すべて	すべて	物理パーティションを仮想パーティションで上書きする機能を実装しました。データウェアハウスの抽出は、作成された仮想パーティションに従ってデータを抽出します。
Amazon Redshift	Amazon Redshift	DML ステートメントでサポートされていない組み込み関数は、プレースホルダーとして NULL に置き換えられます。
Sybase	PostgreSQL	ネイティブ関数のサポートが追加されました。
SAP ASE	MySQL/ Aurora MySQL	Aurora MySQL のデフォルトの独立性レベルは、REPEATABLE READ です。GLOBAL 独立性レベルの設定は、Aurora MySQL ではサポートされていません。セッションスコープのみを変更できます。トランザクションのデフォルトの動作では、REPEATABLE READ と整合性のある読み込みが使用されます。READ COMMITTED で実行するように設計されたアプリケーションは、変更が必要な場合があります。または、デフォルトを READ COMMITTED に明示的に変更することもできます。
SAP ASE	PostgreSQL	拡張パックなしの CONVERT 関数 (オプティミスティック) のサポートが追加されました。
SAP ASE	すべて	システムビュー sysindexes エミュレーションが追加されました。 INTO を指定せずにプロシージャに SELECT 文がある場合、ターゲット上のプロシージャに対して refcursor 型のパラメータ INOUT p_refcur が作成されます。

ソース	Target	新機能、強化点、修正点
Greenplum	Amazon Redshift	CREATE TEMPORARY TABLE を次のように実装しました。

ドキュメント履歴

以下の表には、2018 年 1 月以降に行われた AWS Schema Conversion Tool (AWS SCT) ユーザーガイドに対する重要な変更の説明が記載されています。

RSS フィードにサブスクライブして、このドキュメントの更新に関する通知を受信できます。

変更	説明	日付
AWS SCT ビルド #1.0.672	ビルド 1.0.672 では、Amazon RDS for PostgreSQL 15 をターゲットとして、Microsoft SQL Server バージョン 2022 をソースとしてサポートしています。また、変換されたコードに新しい Amazon Redshift 機能のサポートを追加し、IBM Db2 for z/OS ソースの変換に関する複数の改善を実装し、変換に関する多くの問題を解決します。	2023 年 5 月 8 日
AWS SCT ビルド #1.0.671	ビルド 1.0.671 では、Apache Oozie から AWS Step Functions への移行がサポートされています。また、マルチサーバー評価プロセスのソースとして BigQuery のサポートも追加されています。さらに、ソースとして IBM Db2 for z/OS の新しい変換設定が追加され、変換に関する多くの問題が解決されます。	2023 年 3 月 8 日
AWS SCT ビルド #1.0.670	ビルド 1.0.670 は Hadoop から Amazon EMR への移行をサポートしています。また、	2023 年 1 月 23 日

	マルチサーバー評価プロセスのソースとしての Azure Synapse Analytics のサポートも追加されています。さらに、Java アプリケーションに埋め込まれている SQL コードの変換が改善され、変換に関するさまざまな問題が解決されます。	
AWS SCT ビルド #1.0.669	ビルド 1.0.669 では、Oracle データウェアハウスからのデータ移行のためのネイティブパーティショニングのサポートが実装されています。また、マルチサーバー評価プロセスを改善し、データ抽出エージェントに新機能を追加し、変換に関する多くの問題を解決します。	2022 年 12 月 19 日
AWS SCT ビルド #1.0.668	ビルド 1.0.668 では、Greenplum データベースからのデータ移行のための自動仮想パーティショニングが実装され、Snowflake データベースから Amazon Redshift へのデータ移行のサポートが追加されています。また、C# アプリケーションに埋め込まれている SQL コードの変換が改善され、変換に関する多くの問題が解決されます。	2022 年 11 月 16 日

AWS SCT ビルド #1.0.667	ビルド 1.0.667 は、移行ソースとしての Informatics ETL (抽出、変換、ロード) エンジンをサポートしています。また、拡張パックのバージョンを更新し、Amazon Redshift でサポートされる最小ドライバーバージョンを増やし、変換に関するさまざまな問題を解決します。	2022 年 10 月 13 日
AWS SCT ビルド #1.0.666	ビルド 1.0.666 では MyBatis フレームワークのサポートが追加され、Java アプリケーションの変換が改善されています。また、拡張パックに新しい関数を追加したり、ソースメタデータ・ローダーを強化したり、変換に関するさまざまな問題を解決したりしています。	2022 年 9 月 20 日
AWS SCT ビルド #1.0.665	ビルド 1.0.665 では、移行ソースとして BigQuery がサポートされています。また、新しいバージョンの Babelfish 機能設定ファイルのサポートも実装されています。さらに、データウェアハウスから Amazon Redshift への変換が改善され、変換に関する多くの問題が解決されます。	2022 年 8 月 29 日

AWS SCT ビルド #1.0.664	ビルド 1.0.664 は、移行ソースまたはターゲットとして Amazon Redshift Serverless をサポートします。また、データ抽出タスクに自動メモリバランシングを実装し、AWS SCT で AWS Snowball デバイスに接続できなかったエラーを修正します。さらに、移行ルール内の列の照合順序を変更する機能が追加され、ユーザーインターフェイスが改善され、変換に関する多くの問題が解決されました。	2022 年 7 月 14 日
AWS SCT ビルド #1.0.663	ビルド 1.0.663 では Babelfish for Aurora PostgreSQL 1.2.0 用の Babelfish のサポートが追加され、マルチサーバー評価レポート機能が向上しています。また、移行ルールに新機能が追加され、2 つのローダーエラーが修正され、変換に関する多くの問題が解決されています。	2022 年 6 月 20 日
AWS SCT ビルド #1.0.662	ビルド 1.0.662 は C# アプリケーションに SQL コード変換を実装し、マルチサーバー評価レポートのワークフローを改善しています。また、いくつかの変換上の改善が追加されており、いくつかの変換上の問題が修正されています。	2022 年 5 月 19 日

AWS SCT ビルド #1.0.661	ビルド 1.0.661 では、マイグレーションソースとして IBM Db2 for z/OS がサポートされています。また、抽出、変換、ロード (ETL) スクリプトの AWS Glue Studio への変換に対するサポートも追加されており、いくつかの変換上の問題が修正されています。	2022 年 4 月 21 日
AWS SCT ビルド #1.0.660	ビルド 1.0.660 では、マイグレーションターゲットとして PostgreSQL メジャーバージョン 14 と MariaDB 10.6 がサポートされています。また、マテリアライズドビューの Oracle インデックスの変換のサポートも追加され、変換に関する多くの問題が解決されました。	2022 年 3 月 21 日
AWS SCT ビルド #1.0.659	ビルド 1.0.659 では、マイグレーションターゲットとして Aurora PostgreSQL 互換エディションの PostgreSQL メジャーバージョン 13 がサポートされています。C# アプリケーションに SQL コード変換を実装し、Oracle 統合監査のサポートを追加し、変換に関する多くの問題を解決しています。	2022 年 2 月 21 日

AWS SCT ビルド #1.0.658	ビルド 1.0.658 は Amazon Redshift 仮想ターゲットデータベースプラットフォームと AWS Secrets Manager の統合とサポートの追加を提供します。また、いくつかの変換の改善とバグ修正も行われています。	2022 年 1 月 20 日
AWS SCT ビルド #1.0.657	ビルド 1.0.657 では、Microsoft SQL Server から Aurora PostgreSQL 互換エディション、Amazon RDS for PostgreSQL、およびその他の移行先への変換が改善されています。また、いくつかのユーザーインターフェイスの改善とバグ修正も行われています。	2021 年 12 月 20 日
AWS SCT ビルド #1.0.656	ビルド 1.0.656 は、1 つのプロジェクトで複数のソースデータベースとターゲットデータベースをサポートします。また、変換、最適化戦略、全般的な改善といくつかのバグ修正も追加されています。	2021 年 11 月 22 日
AWS SCT ビルド #1.0.655	ビルド 1.0.655 には、Teradata FastExport ジョブスクリプトの Amazon Redshift RSQL への変換が実装されており、Greenplum でサポートされる最小ドライババージョンが 42.2.5 に増加しています。また、いくつかの改善とバグ修正も行われています。	2021 年 10 月 18 日

AWS SCT ビルド #1.0.654	ビルド 1.0.654 は、シェル、テラデータ高速ロード、テラデータベーシックテラデータクエリ (BTEQ) スクリプトの Amazon Redshift RSQL への変換を実装しています。また、いくつかの変換上の問題が修正され、いくつかの改善とバグ修正も行われています。	2021 年 9 月 16 日
AWS SCT ビルド #1.0.653	ビルド 1.0.653 には、呼び出された関数またはプロシージャで作成された動的 SQL の変換が実装されています。また、暗号化されたルーチンの変換が改善され、いくつかの改善とバグ修正も追加されています。	2021 年 8 月 10 日
AWS SCT ビルド #1.0.652	ビルド 1.0.652 には、コマンドラインインターフェイスのスクリプトコマンドモードと、スキーマ最適化ルールが実装されています。また、いくつかの変換およびパフォーマンス上の改善とバグ修正も行われています。	2021 年 6 月 30 日
AWS SCT ビルド #1.0.651	ビルド 1.0.651 には、いくつかの改善とバグ修正が追加されています。また、AWS Schema Conversion ToolCLI リファレンスの最初のコピーへのアクセスも提供されています。	2021 年 6 月 4 日

AWS SCT ビルド #1.0.650	ビルド 1.0.650 には、ターゲットデータベースとしての Amazon RDS for PostgreSQL 13 へのサポートが実装されており、抽出エージェントも更新されています。また、Microsoft SQL Server、Azure、および Azure Synapse でサポートされている JDBC ドライバーの最小バージョンもアップグレードされています。さらに、いくつかの変換の改善とバグ修正が追加されています。	2021 年 4 月 30 日
AWS SCT ビルド #1.0.649	ビルド 1.0.649 では、ターゲットデータベースとしての MariaDB 10.5 へのサポートが実装されており、Oracle 組み込み関数の変換のための機能も拡張されています。また、いくつかの変換およびパフォーマンス上の改善とバグ修正も行われています。	2021 年 3 月 29 日
AWS SCT ビルド #1.0.648	ビルド 1.0.648 には、いくつかの変換上の改善とバグ修正が追加されています。	2021 年 2 月 22 日

AWS SCT ビルド #1.0.647	ビルド 1.0.647 では、Amazon RDS のデータベースメール機能へのサポートが追加されており、ストレージオブジェクトに対するコメントのロードと変換が実装されています。また、AWS SCT Data Migration Service Assessor と AWS SCT ウィザードが追加されており、ツリーフィルターのユーザーインターフェイスが実装されています。さらに、評価レポートには新たなデザインになったセクションが追加されるとともに、いくつかの改善とバグ修正も追加されています。	2021 年 1 月 15 日
AWS SCT ビルド #1.0.646	ビルド 1.0.646 には、INTERVAL データ型、アイデンティティ列、およびカーソルの変換に対するサポートが追加されており、いくつかの改善とバグ修正も追加されています。	2020 年 12 月 28 日
AWS SCT ビルド #1.0.645	ビルド 1.0.645 には、ETL SSIS から AWS Glue 変換への変換に対するサポートと、いくつかの変換上の改善とバグ修正が追加されています。	2020 年 11 月 16 日
AWS SCT ビルド #1.0.643-1.0.644	ビルド 1.0.644 には、いくつかの変換、パフォーマンス、ユーザーインターフェイス上の改善とバグ修正が追加されています。	2020 年 10 月 14 日

AWS SCT ビルド #1.0.642	ビルド 1.0.642 には、Microsoft SQL Server Integration Service から AWS Glue への変換が実装されており、いくつかの改善とバグ修正も追加されています。	2020 年 8 月 28 日
AWS SCT ビルド #1.0.641	データエクストラクタ向けの SSL サポートが追加されました。また、いくつかの改善とバグ修正も含まれています。	2020 年 7 月 17 日
AWS SCT ビルドs #1.0.633-1.0.640	JDK 8 から Amazon Corretto JDK 11 にアップグレード。他のアップグレード、変更、修正を識別するテーブルが追加されました。	2020年6月22日
AWS WQF 可用性	AWS SCT では、AWS ワークロード資格フレームワーク (AWS WQF) ツールをダウンロードできません。	2020 年 6 月 19 日
AWS SCT ビルドs #1.0.632	SCT UI - スクリプトの適用時に発生するエラーを表示する新しいタブが追加されました。SAP ASE から変換するときに、ソースツリーを SQL として保存できるようになりました。PostgreSQL または Aurora PostgreSQL または Redshift への変換の改善。	2019 年 11 月 19 日

AWS SCT ビルドs #1.0.631 および #1.0.630 (組み合わせ)	Oracle での ROWID のサポート、Microsoft SQL Server と SAP ASE でのシステムオブジェクトのサポートを向上。SQL Server スキーマでの指定子の欠落に対する処理を向上。Greenplum から Redshift への変換のサポートを向上。Amazon Redshift、MariaDB、MySQL、PostgreSQL への移行時のストアドコードの変換のサポートを向上。	2019 年 9 月 30 日
AWS SCT ビルド #1.0.629	Netezza からの変換の場合のストアドプロシージャのサポート。Amazon Redshift、DynamoDB、MySQL、PostgreSQL への変換のサポートを向上。ソースとしての SAP ASE 12.5 のサポートを追加。	2019 年 8 月 20 日
AWS SCT ビルド #1.0.628	DB2、SQL Server、Oracle からの変換のためのサービスエミュレーションのサポート。カーソルやストアドプロシージャのサポート強化など、Amazon Redshift への変換の機能強化。	2019 年 6 月 22 日
AWS SCT ビルド #1.0.627	Amazon Redshift で SQL Server からストアドプロシージャへの変換がサポートされるようになりました。PostgresSQL 11 と MySQL 8.0 への変換に関する機能が強化されました。	2019 年 5 月 31 日

AWS SCT ビルド #1.0.626	PostgreSQL 11 および MySQL 8.0 がターゲットとしてサポートされるようになりました。SAP ASE 15.5 がソースとしてサポートされるようになりました。	2019 年 4 月 26 日
AWS SCT ビルド #1.0.625	アップデートには、Teradata BTEQ を AWS Glue に変換する機能、Oracle 互換モードのサポートによる MariaDB 10.3 への変換のサポート、SAP ASE 15.7 のサポート、および不足している機能をエミュレートするためのサービス置換が含まれます。	2019 年 3 月 25 日
AWS SCT ビルド #1.0.624	アップデートには、Oracle ETL を AWS Glue に変換する機能、Microsoft SQL Server、Oracle、および IBM Db2 LUW から Amazon RDS for MariaDB への変換のサポートが含まれます。また、MySQL の互換性による SAP ASE から RDS for MySQL および Amazon Aurora への変換のサポートも含まれます。Oracle から PostgreSQL への変換時の Orafce 拡張機能のサポートも追加されています。	2019 年 2 月 22 日

AWS SCT ビルド #1.0.623	更新には、SAP ASE データベースを変換する機能、T-SQL スクリプト、DML、および DDL を等価のコードまたはコンポーネントに変換する機能が含まれます。また、変換機能を強化するための、Oracle と Microsoft SQL Server のエミュレーションも追加されています。	2019 年 1 月 25 日
AWS SCT ビルド #1.0.622	アップデートには、データベースおよびアプリケーションの変更など、移行全体のワークロードを分析するワークロード資格フレームワークが含まれています。	2018 年 12 月 20 日
AWS SCT ビルド #1.0.621	更新には、ターゲットとして Aurora PostgreSQL 10 のサポートと、外部テーブルオプションを使用した Netezza からの移行機能が含まれています。	2018 年 11 月 21 日
AWS SCT ビルド #1.0.620	更新には、SQL スクリプトの保存機能、および MySQL への移行の際の Oracle グローバルカーソルのサポートが含まれています。	2018 年 10 月 22 日
AWS SCT ビルド #1.0.619	更新には、Apache Cassandra から DynamoDB への移行のサポートや、ソースとしての Vertica 9 のサポートが含まれます。	2018 年 9 月 20 日

AWS SCT ビルド #1.0.618	更新には、評価レポートの拡張、Oracle ROWID の変換のサポート、SQL Server ユーザー定義テーブルのサポートなどがあります。	2018 年 8 月 24 日
AWS SCT ビルド #1.0.617	更新には、評価レポートの拡張、Oracle ROWID の変換のサポート、SQL Server ユーザー定義テーブルのサポートなどがあります。	2018 年 7 月 24 日
AWS SCT ビルド #1.0.616	更新には、Oracle から Amazon RDS for Oracle への変換、Oracle スケジュールオブジェクトの変換を行う場合の RDS のサポート、および Oracle ジョブ、パーティション化、および Db2 LUW バージョン 10.1 のサポートが含まれます。	2018 年 6 月 26 日
AWS SCT ビルド #1.0.615	更新には、SQL Server から PostgreSQL GOTO ステートメント、PostgreSQL 10 の分割、および Db2 LUW バージョン 10.1 のサポートが含まれています。	2018 年 5 月 24 日
AWS SCT ビルド #1.0.614	更新には Oracle から Oracle DB Links、SQL Server から PostgreSQL インライン関数、および Oracle システムオブジェクトのエミュレーションのサポートが含まれています。	2018 年 4 月 25 日

AWS SCT ビルド #1.0.613	更新には、Db2 LUW、SQL*Plus ファイルの変換、および SQL Server Windows 認証のサポートが含まれています。	2018 年 3 月 28 日
AWS SCT ビルド #1.0.612	更新には、カスタムデータ型のマッピング、Oracle 10 のスキーマ比較、およびグローバル変数の Oracle から PostgreSQL への変換のサポートが含まれています。	2018 年 2 月 22 日
AWS SCT ビルド #1.0.611	更新には、Oracle から PostgreSQL への動的ステートメント、エラーメッセージの選択によるログファイルの起動、およびツリービューでのスキーマの非表示機能のサポートが含まれています。	2018 年 1 月 23 日

以前の更新

以下の表には、2018 年 1 月以前に行われた AWS Schema Conversion Tool (AWS SCT) ユーザーガイドに対する重要な変更の説明が記載されています。

バージョン	変更	説明	変更日
1.0.608	Amazon S3 の FIPS エンドポイントのサポート	連邦情報処理標準 (FIPS) のセキュリティ要件に準拠するために、FIPS エンドポイントを使用して Amazon S3 および Amazon Redshift に接続するように AWS SCT にリクエストできるようになりました。詳細については、「AWS 認証情報の保存」を参照してください。	2017 年 11 月 17 日
1.0.607	Amazon S3 の FIPS エンド	連邦情報処理標準 (FIPS) のセキュリティ要件に準拠するために、FIPS エンドポイン	2017 年 10 月 30 日

バージョン	変更	説明	変更日
	ポイントのサポート	トを使用して Amazon S3 および Amazon Redshift に接続するように AWS SCT にリクエストできるようになりました。詳細については、「AWS 認証情報の保存」を参照してください。	
1.0.607	データ抽出タスクで無視できる LOB	データ抽出タスクを作成する際に、ラージオブジェクト (LOB) を無視することで、抽出するデータの量を減らせるようになりました。詳細については、「AWS SCT データ抽出タスクの作成、実行、監視」を参照してください。	2017 年 10 月 30 日
1.0.605	データ抽出エージェントのタスクログへのアクセス	AWS Schema Conversion Tool ユーザーインターフェイスの便利なリンクから、データ抽出エージェントのタスクログにアクセスできるようになりました。詳細については、「AWS SCT データ抽出タスクの作成、実行、監視」を参照してください。	2017 年 8 月 28 日
1.0.604	コンバーターの強化	AWS Schema Conversion Tool エンジンの機能強化により、異機種間の移行の変換が強化されました。	2017 年 24 月 6 日
1.0.603	データ抽出エージェントがフィルタをサポート	抽出エージェントがデータウェアハウスから抽出するデータをフィルタリングできるようになりました。詳細については、「でのデータ移行ルールの作成 AWS SCT」を参照してください。	2017 年 6 月 16 日

バージョン	変更	説明	変更日
1.0.603	AWS SCT が追加のデータウェアハウスバージョンをサポート	AWS Schema Conversion Tool を使用して、Teradata 13 および Oracle Data Warehouse 10 スキーマを同等の Amazon Redshift スキーマに変換できるようになりました。詳細については、「AWS SCT を使用したデータウェアハウススキーマの Amazon Redshift への変換」を参照してください。	2017 年 6 月 16 日
1.0.602	データ抽出エージェントが追加のデータウェアハウスをサポート	データ抽出エージェントを使用して、Microsoft SQL Server データウェアハウスからデータを抽出できるようになりました。詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。	2017 年 5 月 11 日
1.0.602	データ抽出エージェントがデータをAmazon Redshift にコピー	データ抽出エージェントに、3 つのアップロードモードが導入されました。データを抽出するのみ、データを抽出して Amazon S3 にアップロードするのみ、またはデータを抽出してアップロードし、Amazon Redshift に直接コピーするかを指定できます。詳細については、「AWS SCT データ抽出タスクの作成、実行、監視」を参照してください。	2017 年 5 月 11 日
1.0.601	AWS SCT が追加のデータウェアハウスをサポート	AWS Schema Conversion Tool を使用して、Vertica および Microsoft SQL Server スキーマを同等の Amazon Redshift スキーマに変換できるようになりました。詳細については、「AWS SCT を使用したデータウェアハウススキーマの Amazon Redshift への変換」を参照してください。	2017 年 4 月 18 日

バージョン	変更	説明	変更日
1.0.601	データ抽出エージェントが追加のデータウェアハウスをサポート	データ抽出エージェントを使用して、Greenplum、Netezza、および Vertica データウェアハウスからデータを抽出できるようになりました。詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。	2017 年 4 月 18 日
1.0.601	データ抽出エージェントが追加のオペレーティングシステムをサポート	macOS および Microsoft Windows オペレーティングシステムを実行しているコンピュータに、データ抽出エージェントをインストールできるようになりました。詳細については、「抽出エージェントをインストールする」を参照してください。	2017 年 4 月 18 日
1.0.601	データ抽出エージェントが自動的に Amazon S3 をアップロード	データ抽出エージェントが、抽出されたデータを自動的に Amazon S3 にアップロードするようになりました。詳細については、「データ抽出タスクの出力」を参照してください。	2017 年 4 月 18 日
1.0.600	データ抽出エージェント	データウェアハウスからデータを抽出するデータ抽出エージェントをインストールして、Amazon Redshift で使用する準備ができるようになりました。AWS Schema Conversion Tool を使用してエージェントを登録し、データ抽出タスクを作成します。詳細については、「オンプレミスのデータウェアハウスから Amazon Redshift にデータを移行する」を参照してください。	2017 年 2 月 16 日

バージョン	変更	説明	変更日
1.0.600	お客様からのフィードバック	AWS Schema Conversion Tool に関するフィードバックを提供できるようになりました。バグレポートの提出、機能リクエストの提出、一般情報の提供ができます。詳細については、「フィードバックの提供」を参照してください。	2017 年 2 月 16 日
1.0.502	AWS DMS との統合	AWS Schema Conversion Tool を使用して、AWS DMS エンドポイントやタスクを作成できるようになりました。AWS SCT からこれらのタスクを実行およびモニタリングできます。詳細については、「AWS SCT の AWS DMS との併用」を参照してください。	2016 年 12 月 20 日
1.0.502	ターゲットデータベースとしての PostgreSQL との互換性を持つ Amazon Aurora	AWS Schema Conversion Tool で、ターゲットデータベースとしての PostgreSQL との互換性を持つ Amazon Aurora がサポートされるようになりました。詳細については、「AWS SCT を使用したデータベーススキーマの変換」を参照してください。	2016 年 12 月 20 日
1.0.502	プロファイルのサポート	AWS Schema Conversion Tool に異なるプロファイルを保存して、簡単に切り替えられるようになりました。詳細については、「AWS サービスプロファイルの AWS SCT への保存」を参照してください。	2016 年 12 月 20 日

バージョン	変更	説明	変更日
1.0.501	Greenplum データベースと Netezza のサポート	AWS Schema Conversion Tool を使用して、Greenplum データベースと Netezza から Amazon Redshift にデータウェアハウスのスキーマを変換できるようになりました。詳細については、「AWS SCT を使用したデータウェアハウススキーマの Amazon Redshift への変換」を参照してください。	2016 年 11 月 17 日
1.0.501	Redshift の最適化	AWS Schema Conversion Tool を使用して、Amazon Redshift データベースを最適化できるようになりました。詳細については、「AWS SCT を使用した Amazon Redshift の最適化」を参照してください。	2016 年 11 月 17 日
1.0.500	マッピングのルール	AWS Schema Conversion Tool でスキーマを変換する前に、列のデータ型の変更、あるスキーマから別のスキーマへのオブジェクトの移動、オブジェクトの名前の変更を行うルールを設定できるようになりました。詳細については、「AWS SCT での移行ルールの作成」を参照してください。	2016 年 10 月 4 日
1.0.500	クラウドに移行する	AWS Schema Conversion Tool を使用して、同エンジンを実行する Amazon RDS DB インスタンスに既存のオンプレミスのデータベーススキーマをコピーできます。この機能を使用して、クラウドへの移動やライセンスタイプの変更にかかるコスト削減の可能性を分析できます。詳細については、「AWS SCT を使用した移行評価レポートの作成」を参照してください。	2016 年 10 月 4 日

バージョン	変更	説明	変更日
1.0.400	データウェアハウススキーマの変換	AWS Schema Conversion Tool を使用して、Oracle と Teradata から Amazon Redshift にデータウェアハウスのスキーマを変換できるようになりました。詳細については、「AWS SCT を使用したデータウェアハウススキーマの Amazon Redshift への変換」を参照してください。	2016 年 7 月 13 日
1.0.400	アプリケーション SQL の変換	AWS Schema Conversion Tool を使用して、C++、C#、Java などのアプリケーションコードの SQL コードを変換できるようになりました。詳細については、「AWS SCT を使用したアプリケーション SQL の変換」を参照してください。	2016 年 7 月 13 日
1.0.400	新機能	AWS Schema Conversion Tool に、E メール、ジョブのスケジュール、その他の機能を提供する拡張パックが追加され、AWS Lambda 関数と Python ライブラリのインストール、作成、設定に役立つウィザードも追加されました。詳細については、AWS LambdaAWS SCT エクステンションパックの関数を使用する および AWS SCT 拡張パックにカスタムライブラリを使用する を参照してください。	2016 年 7 月 13 日
1.0.301	SSL サポート	AWS Schema Conversion Tool の使用時、Secure Sockets Layer (SSL) でソースデータベースに接続できるようになりました。	2016 年 5 月 19 日
1.0.203	新機能	変換対象のソースデータベースとして MySQL と PostgreSQL がサポートされるようになりました。	2016 年 4 月 11 日

バージョン	変更	説明	変更日
1.0.202	メンテナンスリリース	ターゲットデータベースエンジン用に変換された SQL の編集がサポートされるようになりました。ソースデータベースおよびターゲット DB インスタンスツリービューでの選択機能が改良されました。Transparent Network Substrate (TNS) 名を使用した Oracle ソースデータベースへの接続がサポートされるようになりました。	2016 年 3 月 2 日
1.0.200	メンテナンスリリース	ターゲットデータベースエンジンとして PostgreSQL がサポートされるようになりました。変換されたスキーマをスクリプトとして生成し、ターゲット DB インスタンスへのスキーマの適用前にファイルにスクリプトを保存する機能が追加されました。	2016 年 1 月 14 日
1.0.103	メンテナンスリリース	オフラインプロジェクト機能、新しいバージョンの確認機能、メモリとパフォーマンスの管理機能が追加されました。	2015 年 12 月 2 日
1.0.101	メンテナンスリリース	[Create New Database Migration Project] (データベース移行レポートを作成) ウィザードが追加されました。データベース移行評価レポートを PDF ファイルとして保存する機能が追加されました。	2015 年 10 月 19 日
1.0.100	プレビューリリース	AWS Schema Conversion Tool プレビューリリースのユーザーガイドです。	2015 年 10 月 7 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。