AWS Schema Conversion Tool
User Guide
Version 1.0
# Table of Contents

- What Is the AWS Schema Conversion Tool? ........................................................................................................ 1
  - Schema Conversion Overview .......................................................................................................................... 2
  - Feedback ......................................................................................................................................................... 2
- Installing, Verifying, and Updating .......................................................................................................................... 4
  - Installing the AWS SCT .................................................................................................................................. 4
    - Installing Previous Versions .......................................................................................................................... 5
  - Verifying the AWS SCT File Download ........................................................................................................... 5
    - Verifying the Checksum of the AWS SCT File ............................................................................................... 6
    - Verifying the AWS SCT RPM Files on Fedora ............................................................................................. 6
    - Verifying the AWS SCT DEB Files on Ubuntu ............................................................................................. 7
    - Verifying the AWS SCT MSI File on Microsoft Windows ........................................................................ 7
    - Verifying the AWS SCT Application on Mac OS .......................................................................................... 7
  - Installing the Required Database Drivers ....................................................................................................... 8
    - Installing JDBC Drivers on Linux .................................................................................................................. 9
    - Storing Driver Paths in the Global Settings ................................................................................................. 10
- Using the AWS SCT User Interface ........................................................................................................................... 12
  - The AWS SCT Project Window ....................................................................................................................... 12
  - Storing AWS Profiles ...................................................................................................................................... 13
    - Storing AWS Credentials ........................................................................................................................... 14
    - Setting the Default Profile for a Project ...................................................................................................... 16
  - Storing Database Passwords ........................................................................................................................... 16
  - AWS SCT Tree Filters .................................................................................................................................... 16
    - Importing a File List for the Tree Filter ....................................................................................................... 18
    - Hiding Schemas ........................................................................................................................................... 18
  - Keyboard Shortcuts ......................................................................................................................................... 19
  - The Database Migration Assessment Report ..................................................................................................... 20
  - Starting SCT ................................................................................................................................................... 23
  - Creating a Project ............................................................................................................................................ 24
  - Converting Your Schema ................................................................................................................................. 25
  - Applying the Converted Schema to Your Target DB Instance ......................................................................... 26
- Getting Started ................................................................................................................................................... 28
  - Source Databases for AWS SCT ........................................................................................................................ 29
    - Using Oracle as a Source for AWS Schema Conversion Tool (AWS SCT) .................................................. 29
      - Permissions Needed for Oracle as a Source ............................................................................................... 30
      - Connecting to Oracle as a Source .............................................................................................................. 30
      - Oracle to PostgreSQL .................................................................................................................................. 33
      - Oracle to MySQL ......................................................................................................................................... 35
      - Oracle to Amazon RDS Oracle .................................................................................................................... 36
    - Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT) ......................... 37
      - Source: Microsoft SQL Server .................................................................................................................... 38
      - Source: Windows Authentication with Microsoft SQL Server ...................................................................... 39
      - Connecting to SQL Server as a Source ....................................................................................................... 40
      - SQL Server to MySQL ................................................................................................................................. 42
      - SQL Server to PostgreSQL .......................................................................................................................... 43
      - SQL Server to Amazon RDS SQL Server .................................................................................................... 45
    - Using MySQL as a Source for AWS Schema Conversion Tool (AWS SCT) ................................................. 45
      - Source: MySQL ............................................................................................................................................ 45
      - Connecting to MySQL as a Source ............................................................................................................. 46
    - Using PostgreSQL as a Source for AWS Schema Conversion Tool (AWS SCT) ........................................... 47
      - Source: PostgreSQL ................................................................................................................................... 48
      - Connecting to PostgreSQL as a Source ..................................................................................................... 48
    - Using Db2 LUW as a Source for AWS Schema Conversion Tool (AWS SCT) .............................................. 49

Version 1.0

iii
What Is the AWS Schema Conversion Tool?

You can use the AWS Schema Conversion Tool (AWS SCT) to convert your existing database schema from one database engine to another. You can convert relational OLTP schema, or data warehouse schema. Your converted schema is suitable for an Amazon Relational Database Service (Amazon RDS) MySQL DB instance, an Amazon Aurora DB cluster, an Amazon RDS PostgreSQL DB instance, or an Amazon Redshift cluster. The converted schema can also be used with a database on an Amazon EC2 instance or stored as data on an Amazon S3 bucket.

AWS SCT supports several industry standards, including Federal Information Processing Standards (FIPS) when connecting to an Amazon S3 bucket or another AWS resource. AWS SCT is also compliant with Federal Risk and Authorization Management Program (FedRAMP).

AWS SCT supports the following OLTP conversions.

<table>
<thead>
<tr>
<th>Source Database</th>
<th>Target Database on Amazon RDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server (version 2008 and later)</td>
<td>Amazon Aurora (MySQL or PostgreSQL), Microsoft SQL Server, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>MySQL (version 5.5 and later)</td>
<td>Amazon Aurora (PostgreSQL), MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>You can migrate schema and data from MySQL to an Amazon Aurora (MySQL) DB cluster without using AWS SCT. For more information, see Migrating Data to an Amazon Aurora DB Cluster.</td>
</tr>
<tr>
<td>Oracle (version 10.2 and later)</td>
<td>Amazon Aurora (MySQL or PostgreSQL), MySQL, Oracle, PostgreSQL</td>
</tr>
<tr>
<td>PostgreSQL (version 9.1 and later)</td>
<td>Amazon Aurora (MySQL), MySQL, PostgreSQL</td>
</tr>
<tr>
<td>IBM Db2 LUW (versions 9.1, 9.5, 9.7, 10.5, and 11.1)</td>
<td>Amazon Aurora (MySQL), MySQL, PostgreSQL</td>
</tr>
</tbody>
</table>

AWS SCT supports the following data warehouse conversions.

<table>
<thead>
<tr>
<th>Source Database</th>
<th>Target Database on Amazon Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greenplum Database (version 4.3 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Microsoft SQL Server (version 2008 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Netezza (version 7.0.3 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Oracle (version 10 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Teradata (version 13 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Vertica (version 7.2.2 and later)</td>
<td>Amazon Redshift</td>
</tr>
</tbody>
</table>
Schema Conversion Overview

AWS SCT provides a project-based user interface to automatically convert the database schema of your source database into a format compatible with your target Amazon RDS instance. If schema from your source database can't be converted automatically, AWS SCT provides guidance on how you can create equivalent schema in your target Amazon RDS database.

For information about how to install AWS SCT, see Installing, Verifying, and Updating the AWS Schema Conversion Tool (AWS SCT) (p. 4).

For an introduction to the AWS SCT user interface, see Using the AWS Schema Conversion Tool (AWS SCT) User Interface (p. 12).

For information on the conversion process, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 70).

In addition to converting your existing database schema from one database engine to another, AWS SCT has some additional features that help you move your data and applications to the AWS Cloud:

- You can use data extraction agents to extract data from your data warehouse to prepare to migrate it to Amazon Redshift. To manage the data extraction agents, you can use AWS SCT. For more information, see Using Data Extraction Agents (p. 110).

- You can use AWS SCT to create AWS DMS endpoints and tasks. You can run and monitor these tasks from AWS SCT. For more information, see Using the AWS Schema Conversion Tool with the AWS Database Migration Service (p. 109).

- In some cases, database features can't be converted to equivalent Amazon RDS or Amazon Redshift features. The AWS SCT extension pack wizard can help you install AWS Lambda functions and Python libraries to emulate the features that can't be converted. For more information, see Using the AWS Schema Conversion Tool Extension Pack (p. 140).

- You can use AWS SCT to optimize your existing Amazon Redshift database. AWS SCT recommends sort keys and distribution keys to optimize your database. For more information, see Optimizing Amazon Redshift by Using the AWS Schema Conversion Tool (p. 107).

- You can use AWS SCT to copy your existing on-premises database schema to an Amazon RDS DB instance running the same engine. You can use this feature to analyze potential cost savings of moving to the cloud and of changing your license type.

- You can use AWS SCT to convert SQL in your C++, C#, Java, or other application code. You can view, analyze, edit, and save the converted SQL code. For more information, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 134).

Providing Customer Feedback

You can provide feedback about the AWS Schema Conversion Tool. You can file a bug report, you can submit a feature request, or you can provide general information.

To provide feedback about AWS SCT.

1. Start the AWS Schema Conversion Tool.
2. Open the Help menu and then choose Leave Feedback. The Leave Feedback dialog box appears.
3. For Area, choose Information, Bug report, or Feature request.
4. For Source database, choose your source database. Choose Any if your feedback is not specific to a particular database.
5. For Target database, choose your target database. Choose Any if your feedback is not specific to a particular database.
6. For Title, type a title for your feedback.
7. For Message, type your feedback.
8. Choose Send to submit your feedback.
Installing, Verifying, and Updating the AWS Schema Conversion Tool (AWS SCT)

The AWS Schema Conversion Tool (AWS SCT) is a standalone application that provides a project-based user interface. AWS SCT is available for Fedora Linux, macOS, Microsoft Windows, and Ubuntu Linux version 15.04. AWS SCT is supported only on 64-bit operating systems. AWS SCT also installs the Java Runtime Environment (JRE) version 8u45.

To ensure that you get the correct version of the AWS SCT distribution file, we provide verification steps after you download the compressed file. You can verify the file using the steps provided.

Topics
- Installing the AWS SCT (p. 4)
- Verifying the AWS SCT File Download (p. 5)
- Installing the Required Database Drivers (p. 8)
- Updating the AWS SCT (p. 11)

Installing the AWS SCT

To install the AWS SCT

1. Download the compressed file that contains the AWS SCT installer, using the link for your operating system. All compressed files have a .zip extension. When you extract the AWS SCT installer file, it will be in the appropriate format for your operating system.

   - Microsoft Windows
   - Apple macOS
   - Ubuntu Linux (.deb)
   - Fedora Linux (.rpm)

2. Extract the AWS SCT installer file for your operating system, shown following.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora Linux</td>
<td>aws-schema-conversion-tool-1.0.build-number.x86_64.rpm</td>
</tr>
<tr>
<td>macOS</td>
<td>AWS Schema Conversion Tool-1.0.build-number.dmg</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>AWS Schema Conversion Tool-1.0.build-number.msi</td>
</tr>
<tr>
<td>Ubuntu Linux</td>
<td>aws-schema-conversion-tool-1.0.build-number.deb</td>
</tr>
</tbody>
</table>

3. Run the AWS SCT installer file extracted in the previous step. Use the instructions for your operating system, shown following.
### Installing Previous Versions of the AWS SCT

You can download and install previous versions of the AWS SCT. Use the following format to download a previous version. You must provide the version and OS information using this format.

```
https://d211wdu1froga6.cloudfront.net/builds/1.0/<version>/<OS>/aws-schema-conversion-tool-1.0.zip
```

For example, to download AWS SCT version 607, do the following:

- MacOS - [https://d211wdu1froga6.cloudfront.net/builds/1.0/607/MacOS/aws-schema-conversion-tool-1.0.zip](https://d211wdu1froga6.cloudfront.net/builds/1.0/607/MacOS/aws-schema-conversion-tool-1.0.zip)
- Windows - [https://d211wdu1froga6.cloudfront.net/builds/1.0/607/Windows/aws-schema-conversion-tool-1.0.zip](https://d211wdu1froga6.cloudfront.net/builds/1.0/607/Windows/aws-schema-conversion-tool-1.0.zip)
- Ubuntu - [https://d211wdu1froga6.cloudfront.net/builds/1.0/607/Ubuntu/aws-schema-conversion-tool-1.0.zip](https://d211wdu1froga6.cloudfront.net/builds/1.0/607/Ubuntu/aws-schema-conversion-tool-1.0.zip)
- Fedora - [https://d211wdu1froga6.cloudfront.net/builds/1.0/607/Fedora/aws-schema-conversion-tool-1.0.zip](https://d211wdu1froga6.cloudfront.net/builds/1.0/607/Fedora/aws-schema-conversion-tool-1.0.zip)

### Verifying the AWS SCT File Download

There are several ways you can verify the distribution file of the AWS SCT. The simplest is to compare the checksum of the file with the published checksum from AWS. As an additional level of security, you can use the procedures below to verify the distribution file, based on the operating system where you installed the file.

This section includes the following topics.
Verifying the Checksum of the AWS SCT File

In order to detect any errors that could have been introduced when downloading or storing the AWS SCT compressed file, you can compare the file checksum with a value provided by AWS. AWS uses the SHA256 algorithm for the checksum.

To verify the AWS SCT distribution file using a checksum

1. Download the AWS SCT distribution file using the links in the Installing section.
2. Download the latest checksum file, called sha256Check.txt. For example, the file can appear like the following:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora</td>
<td>b4f5f66f91bfcc1b312e2827e960691c269a9002cd1371cf1841593f88cbb5e6</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>4315eb664494fcd95932351f00399adb6c6cf64b9f30adda2eece903c54eca4</td>
</tr>
<tr>
<td>Windows</td>
<td>6e29679a4c35c5396a06dd50f308981e4ec34bd0acd608874470700a0ae9a23</td>
</tr>
<tr>
<td>MacOs</td>
<td>ed5d3ab49309e92ac4d2ef439d35449ac1326f470c23dc5866e1bf0a60b0e67</td>
</tr>
</tbody>
</table>
3. Run the SHA256 validation command for your operating system in the directory that contains the distribution file. For example, the command to run on the Mac operating system is the following:

   shasum -a 256 aws-schema-conversion-tool-1.0.latest.zip
4. Compare the results of the command with the value shown in the sha256Check.txt file. The two values should match.

Verifying the AWS SCT RPM Files on Fedora

AWS provides another level of validation in addition to the distribution file checksum. All RPM files in the distribution file are signed by an AWS private key. The public GPG key can be viewed at amazon.com.public.gpg-key.

To verify the AWS SCT RPM files on Fedora

1. Download the AWS SCT distribution file using the links in the Installing section.
2. Verifying the checksum of the AWS SCT distribution file.
3. Extract the contents of the distribution file. Locate the RPM file you want to verify.
4. Download GPG public key from amazon.com.public.gpg-key
5. Import the public key to your RPM DB (make sure you have the appropriate permissions) by using the following command:

   sudo rpm --import aws-dms-team@amazon.com.public.gpg-key
6. Check that the import was successful by running the following command:
Verifying the AWS SCT DEB Files on Ubuntu

AWS provides another level of validation in addition to the distribution file checksum. All DEB files in the distribution file are signed by a GPG detached signature.

**To verify the AWS SCT DEB files on Ubuntu**

1. Download the AWS SCT distribution file using the links in the Installing section.
2. Verifying the checksum of the AWS SCT distribution file.
3. Extract the contents of the distribution file. Locate the DEB file you want to verify.
4. Download the detached signature from *aws-schema-conversion-tool-1.0.latest.deb.asc*.
5. Download the GPG public key from *amazon.com.public.gpg-key*.
6. Import the GPG public key by running the following command:
   ```bash
gpg --import aws-dms-team@amazon.com.public.gpg-key
   ```
7. Verify the signature by running the following command:
   ```bash
gpg --verify aws-schema-conversion-tool-1.0.latest.deb.asc aws-schema-conversion-tool-1.0.build number-1.x86_64.rpm
   ```

Verifying the AWS SCT MSI File on Microsoft Windows

AWS provides another level of validation in addition to the distribution file checksum. The MSI file has a digital signature you can check to ensure it was signed by AWS.

**To verify the AWS SCT MSI file on Windows**

1. Download the AWS SCT distribution file using the links in the Installing section.
2. Verifying the checksum of the AWS SCT distribution file.
3. Extract the contents of the distribution file. Locate the MSI file you want to verify.
4. In Windows Explorer, right-click the MSI file and select **Properties**.
5. Choose the **Digital Signatures** tab.
6. Verify that the digital signature is from Amazon Services LLC.

Verifying the AWS SCT Application on Mac OS

AWS provides another level of validation in addition to the distribution file checksum. Once you have installed the AWS SCT on the Mac OS, you can verify the application using the following procedure.
To verify the AWS SCT Application on Mac OS

1. Download the AWS SCT distribution file using the links in the Installing section.
2. Verifying the checksum of the AWS SCT distribution file.
3. Extract the contents of the distribution file.
4. Double-click the DMG file.
5. Install the AWS SCT.
6. Verify the application by running the following command:

   codesign -dvvv /Applications/AWS\ Schema\ Conversion\ Tool.app/

Installing the Required Database Drivers

For the AWS SCT to work correctly, you must install the JDBC drivers for your source and target database engines.

After you download the drivers, you give the location of the driver files. For more information, see Storing Driver Paths in the Global Settings (p. 10).

You can download the database drivers from the following locations.

**Important**
Install the latest version of the driver available. The versions in the table following are example version numbers.

<table>
<thead>
<tr>
<th>Database Engine</th>
<th>Drivers</th>
<th>Download Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Aurora (MySQL compatible)</td>
<td>mysql-connector-java-5.1.6.jar</td>
<td><a href="https://www.mysql.com/products/connector/">https://www.mysql.com/products/connector/</a></td>
</tr>
<tr>
<td>Amazon Aurora (PostgreSQL compatible)</td>
<td>postgresql-9.4-1204-jdbc42.jar</td>
<td><a href="https://jdbc.postgresql.org/download.html">https://jdbc.postgresql.org/download.html</a></td>
</tr>
<tr>
<td>Amazon Redshift</td>
<td>RedshiftJDBC41-1.1.10.1010.jar</td>
<td><a href="http://docs.aws.amazon.com/redshift/latest/mgmt/configure-jdbc-connection.html">http://docs.aws.amazon.com/redshift/latest/mgmt/configure-jdbc-connection.html</a></td>
</tr>
<tr>
<td>Greenplum Database</td>
<td>postgresql-9.4-1204-jdbc42.jar</td>
<td><a href="https://jdbc.postgresql.org/">https://jdbc.postgresql.org/</a></td>
</tr>
<tr>
<td>MySQL</td>
<td>mysql-connector-java-5.1.6.jar</td>
<td><a href="https://www.mysql.com/products/connector/">https://www.mysql.com/products/connector/</a></td>
</tr>
</tbody>
</table>
Installing JDBC Drivers on Linux

You can use the following steps to install the JDBC drivers on your Linux system for use with AWS SCT.

**To install JDBC drivers on your Linux system**

1. Create a directory to store the JDBC drivers in.

   ```bash
   PROMPT$ sudo mkdir -p /usr/local/jdbc-drivers
   ```

2. Install the JDBC driver for your database engine using the commands shown following.

### Database Engine | Installation Commands

| **Amazon Aurora (MySQL compatible)** | PROMPT$ cd /usr/local/jdbc-drivers
PROMPT$ sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz |
| **Amazon Aurora (PostgreSQL compatible)** | 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/ojdbc7.jar . |
| **PostgreSQL** | PROMPT$ cd /usr/local/jdbc-drivers |

### Download Location

- **Oracle**
  - `ojdbc7.jar`
  - [Download](http://www.oracle.com/technetwork/database/features/jdbc-drivers-12c-download-1958347.html)
- **PostgreSQL**
  - `postgresql-9.4-1204-jdbc42.jar`
  - [Download](https://jdbc.postgresql.org/download.html)
- **Teradata**
  - `terajdbc4.jar`
  - `tdgssconfig.jar`
  - [Download](https://downloads.teradata.com/download/connectivity/jdbc-driver)
- **Vertica**
  - `vertica-jdbc-7.2.3-0_all`
  - [Download](https://my.vertica.com/download/vertica/client-drivers/)

### Additional Information

Driver versions 7 and later are supported.
Storing Driver Paths in the Global Settings

After you have downloaded and installed the required JDBC drivers, you can set the location of the drivers globally in the AWS SCT settings. If you don't set the location of the drivers globally, the application asks you for the location of the drivers when you connect to a database.

**To update the driver file locations**

1. In the AWS SCT, choose **Settings**, and then choose **Global Settings**.

2. For **Global settings**, choose **Drivers**. Add the file path to the JDBC driver for your source database engine and your target Amazon RDS DB instance database engine.

   **Note**
   For Teradata, you specify two drivers separated by a semicolon.

3. When you are finished adding the driver paths, choose **OK**.
Updating the AWS SCT

AWS periodically updates the AWS SCT with new features and functionality. If you are updating from a previous version, create a new AWS SCT project and reconver any database objects you are using.

You can check to see if updates exist for the AWS SCT.

**To check for updates to AWS SCT**

1. When in the AWS SCT, choose Help and then choose Check for Updates.
2. In the **Check for Updates dialog box**, choose What's New. If the link does not appear, you have the latest version.
Using the AWS Schema Conversion Tool (AWS SCT) User Interface

The following sections help you work with the AWS SCT user interface. For information on installing AWS SCT, see Installing, Verifying, and Updating the AWS Schema Conversion Tool (AWS SCT) (p. 4).

Topics

- The AWS SCT Project Window (p. 12)
- Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13)
- Storing Database Passwords (p. 16)
- Using AWS SCT Tree Filters (p. 16)
- Hiding Schemas in the AWS SCT Tree View (p. 18)
- Keyboard Shortcuts for the AWS SCT (p. 19)
- Creating and Reviewing the Database Migration Assessment Report (p. 20)
- Starting the AWS Schema Conversion Tool (p. 23)
- Creating an AWS Schema Conversion Tool Project (p. 24)
- Converting Your Schema (p. 25)
- Applying the Converted Schema to Your Target DB Instance (p. 26)

The AWS SCT Project Window

The illustration following is what you see in the AWS SCT when you create a schema migration project, and then convert a schema.

1. In the left panel, the schema from your source database is presented in a tree view. Your database schema is "lazy loaded." In other words, when you select an item from the tree view, AWS SCT gets and displays the current schema from your source database.

2. In the top middle panel, action items appear for schema elements from the source database engine that couldn’t be converted automatically to the target database engine.

3. In the right panel, the schema from your target DB instance is presented in a tree view. Your database schema is "lazy loaded." That is, at the point when you select an item from the tree view, AWS SCT gets and displays the current schema from your target database.
4. In the lower left panel, when you choose a schema element, properties describing the source schema element and the SQL command to create that element in the source database are displayed.

5. In the lower right panel, when you choose a schema element, properties describing the target schema element and the SQL command to create that element in the target database are displayed. You can edit this SQL command and save the updated command with your project.

Using AWS Service Profiles in the AWS Schema Conversion Tool

You can store your AWS credentials in the AWS Schema Conversion Tool (AWS SCT). AWS SCT uses your credentials when you use features that integrate with AWS services. For example, AWS SCT integrates with Amazon S3, AWS Lambda, Amazon Relational Database Service, and AWS Database Migration Service.

AWS SCT asks you for your AWS credentials when you access a feature that requires them. You can store your credentials in the global application settings. When AWS SCT asks for your credentials, you can select the stored credentials.
You can store different sets of AWS credentials in the global application settings. For example, you can store one set of credentials that you use in test scenarios, and a different set of credentials that you use in production scenarios. You can also store different credentials for different AWS Regions.

**Storing AWS Credentials**

Use the following procedure to store AWS credentials globally.

**To store AWS credentials**

1. Start the AWS Schema Conversion Tool.
2. Open the **Settings Menu**, and then choose **Global Settings**. The **Global Settings** dialog box appears.

Choose **AWS Service Profiles**, as shown following.

![AWS Service Profiles](image)

3. Choose **Add new AWS Service Profile**.
4. Enter your AWS information as follows.

<table>
<thead>
<tr>
<th>AWS SCT Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema/Database Name</strong></td>
<td>Choose the schema and database name that you want to filter on. For schema name, you can enter “%” to select all schemas.</td>
</tr>
</tbody>
</table>

a. For **Profile name**, type a name for your profile.
b. For **AWS Access Key**, type your AWS access key.
c. For **AWS Secret Key**, type your AWS secret key.
d. For **Region**, choose the region for your profile.
e. For **S3 Bucket**, choose the Amazon S3 bucket for your profile. You need to specify a bucket only if you are using a feature that connects to S3.

f. Choose **Use FIPS endpoint for S3** if you need to comply with the security requirements for the Federal Information Processing Standard (FIPS). FIPS endpoints are available in the following AWS Regions:
   - US East (N. Virginia) Region
   - US East (Ohio) Region
   - US West (N. California) Region
   - US West (Oregon) Region

5. Choose **Test Connection** to verify that your credentials are correct and active.

The **Test Connection** dialog box appears. You can see the status for each of the services connected to your profile. **Pass** indicates that the profile can successfully access the service.

![Test Connection dialog box]

6. After you have configured your profile, choose **Save** to save your profile or **Cancel** to cancel your changes.

7. Choose **OK** to close the **Global Settings** dialog box.
Setting the Default Profile for a Project

You can set the default profile for an AWS SCT project. Doing this associates the AWS credentials stored in the profile with the project. With your project open, use the following procedure to set the default profile.

To set the default profile for a project

1. Start the AWS Schema Conversion Tool.
2. Open the Settings Menu, and then choose Project Settings. The Current project settings dialog box appears.
3. Choose the Project Environment tab.
4. For AWS Service Profile, choose the profile that you want to associate with the project.
5. Choose OK to close the Current project settings dialog box. You can also choose Cancel to cancel your changes.

Storing Database Passwords

You can store a database password or SSL certificate in the AWS SCT cache. To store a password, choose Store Password when you create a connection.

The password is encrypted using the randomly generated token in the seed.dat file. The password is then stored with the user name in the cache file. If you lose the seed.dat file or it becomes corrupted, the database password might be unencrypted incorrectly. In this case, the connection fails.

Using AWS SCT Tree Filters

To migrate data from a source to a target, AWS SCT loads all metadata from source and target databases into a tree structure. This structure appears in AWS SCT as the tree view in the main project window.

Some databases can have a large number of objects in the tree structure. You can use tree filters in AWS SCT to search for objects in the source and target tree structures. When you use a tree filter, you don't change the objects that are converted when you convert your database. The filter only changes what you see in the tree.

Tree filters work with objects that AWS SCT has preloaded. In other words, AWS SCT doesn't load objects from the database during searches. This approach means that the tree structure generally contains fewer objects than are present in the database.

For tree filters, keep the following in mind:

- The filter default is ANY, which means that the filter uses a name search to find objects.
- When you select one or more object types, you see only those types of objects in the tree.
- You can use the filter mask to show different types of symbols, including Unicode, spaces, and special characters. The “%” character is the wildcard for any symbol.
- After you apply a filter, the count shows only the number of filtered objects.

To create a tree filter

1. Open an existing AWS SCT project.
2. Connect to the database you want to apply the tree filter to.
3. Choose the filter icon.

**Note**
The undo filter icon is grayed out because no filter is currently applied.

4. Enter the following information in the **Tree Filter** dialog box. Options in the dialog box are different for each database engine.

<table>
<thead>
<tr>
<th>AWS SCT Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema/Database Name</strong></td>
<td>Choose the schema and database name you want to filter on. For schema name, you can enter “%” to choose all schemas.</td>
</tr>
<tr>
<td><strong>Object Name</strong></td>
<td>Choose the object name if you want to search by object name.</td>
</tr>
<tr>
<td><strong>Tree States</strong></td>
<td>Choose the table state. The options include the following:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Any</strong> – Show all tables.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Converted</strong> – Show only converted tables.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Not Converted</strong> – Show only tables that have not been converted.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Has Actions</strong> – Show tables with pending actions.</td>
</tr>
<tr>
<td><strong>Object types list</strong></td>
<td>Choose objects from the list of object types that you can filter on. Expand the tree nodes to load more objects to the object types list.</td>
</tr>
</tbody>
</table>
5. Choose **Apply**. After you choose **Apply**, the undo filter icon (next to the filter icon) is enabled. Use this icon if you want to remove the filters you applied.

6. Choose **Close** to close the dialog box.

When you filter the schema that appears in the tree, you don't change the objects that are converted when you convert your schema. The filter only changes what you see in the tree.

**Importing a File List for the Tree Filter**

You can import a file that contains names or values that you want the tree filter to use. In this file, the following convention is used:

- **Object** is the type of object that you want to find.
- **Database** is the name of database where this object exists.
- **Schema** is the name of schema where this object exists.
- **Name** is the object name.

The file to import should have the following format:

- **Object;Database;Schema;Name** – This format is mandatory for the Microsoft SQL Server, SQL Data Warehouse, and Netezza dialects of SQL.
- **Object;Schema;Name** – Use this format for other SQL dialects.

**To import a file list for the tree filter**

1. Open an existing AWS SCT project, connect to the database you want to apply the tree filter to, and then choose the filter icon.

2. Choose the **Import File List** tab.

3. Choose **Import File**.

4. Choose a file to import, and then choose **Open**.

5. Choose **Apply**, and then choose **Close**.

**Hiding Schemas in the AWS SCT Tree View**

By using tree view settings, you specify what schemas and databases you want to see in the AWS SCT tree view. You can hide empty schemas, empty databases, system databases, and user-defined databases and schemas.

**To hide databases and schemas in tree view**

1. Open an AWS SCT project.

2. Connect to the data store that you want to show in tree view.

3. Choose **Settings, Global Settings, Tree View**.
4. In the **Tree View Settings** section, do the following:

   - For **Hide System Databases/Schemas**, choose system databases and schemas by name to hide them.
   - For **Hide User Defined Databases/Schemas**, type the names of user-defined schemas and databases that you want to hide, and then choose **Add**. The names are case insensitive.
   - Choose **Reset to Default** to reset the tree view to default settings.

5. Choose **OK**.

### Keyboard Shortcuts for the AWS SCT

The following are the keyboard shortcuts that you can use with the AWS SCT.

<table>
<thead>
<tr>
<th>Keyboard Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+N</td>
<td>Create a new project.</td>
</tr>
<tr>
<td>Ctrl+O</td>
<td>Open an existing project.</td>
</tr>
<tr>
<td>Ctrl+S</td>
<td>Save an open project.</td>
</tr>
<tr>
<td>Ctrl+W</td>
<td>Create a new project by using the wizard.</td>
</tr>
<tr>
<td>Ctrl+L</td>
<td>Connect to the source database.</td>
</tr>
<tr>
<td>Ctrl+R</td>
<td>Connect to the target database.</td>
</tr>
</tbody>
</table>
Creating and Reviewing the Database Migration Assessment Report

The database migration assessment report summarizes all of the action items for schema that can't be converted automatically to the engine of your target Amazon RDS DB instance. The report also includes estimates of the amount of effort that it will take to write the equivalent code for your target DB instance.

You can create (or update) a database migration assessment report in your project at any time by using the following procedure.

To create and view the database migration assessment report

1. In the left panel that displays the schema from your source database, choose a schema object to create an assessment report for. Open the context (right-click) menu for the object, and then choose Create Report.

The assessment report view opens.

2. Choose the Action Items tab.

The Action Items tab displays a list of items that describe the schema that can't be converted automatically. Select one of the action items from the list. AWS SCT highlights the item from your schema that the action item applies to, as shown following.
3. Choose the **Summary** tab.

The **Summary** tab displays the summary information from the database migration assessment report. It shows the number of items that were converted automatically, and the number of items that were not converted automatically. The summary also includes an estimate of the time that it will take to create schema in your target DB instance that are equivalent to those in your source database.

The section **License Evaluation and Cloud Support** contains information about moving your existing on-premises database schema to an Amazon RDS DB instance running the same engine. For example, if you want to change license types, this section of the report tells you which features from your current database should be removed.

An example of an assessment report summary is shown following.
4. Choose the **Summary** tab, and then choose **Save to PDF**. The database migration assessment report is saved as a PDF file. The PDF file contains both the summary and action item information.

You can also choose **Save to CSV** to save the report as a comma-separated values (CSV) file. The CSV file contains only action item information.
Starting the AWS Schema Conversion Tool

To start the AWS Schema Conversion Tool, use the instructions for your operating system shown following.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora Linux</td>
<td>Run the following command:</td>
</tr>
<tr>
<td></td>
<td>/opt/AWSSchemaConversionTool/AWSSchemaConversionTool</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>Double-click the icon for the application.</td>
</tr>
<tr>
<td>Ubuntu Linux</td>
<td>Run the following command:</td>
</tr>
</tbody>
</table>

Database Objects with Conversion Actions for MySQL

Of the total 179 database storage object(s) in the source database, we were able to identify 169 (94%) database storage object(s) that can be converted automatically or with minimal changes to MySQL.

10 (5%) database storage object(s) required 58 medium and 10 significant user action(s) to complete the conversion.

Figure: Conversion statistics for database storage objects

Detailed Recommendations for MySQL Migrations

If you choose to migrate your SQL Server database to MySQL, we recommend the following actions.

Storage Object Actions

Constraint Changes

Some changes are required to CONSTRAINTs that cannot be converted automatically. You'll need to address these issues manually.
Creating an AWS Schema Conversion Tool Project

The following procedure shows you how to create an AWS Schema Conversion Tool project.

**To create your project**

1. Start the AWS Schema Conversion Tool.
2. Choose **New Project** from the File menu. The **New Project** dialog box appears.

3. Add the following preliminary project information.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>Type a name for your project, which is stored locally on your computer.</td>
</tr>
<tr>
<td>Location</td>
<td>Type the location for your local project file.</td>
</tr>
<tr>
<td>Source DB Engine</td>
<td>Choose <strong>Transactional Database (OLTP)</strong> or <strong>Data Warehouse (OLAP)</strong>.</td>
</tr>
<tr>
<td>Target DB Engine</td>
<td>(OLTP) Choose Microsoft SQL Server, MySQL, Oracle, or PostgreSQL. (OLAP) Choose Amazon Redshift, Greenplum, Microsoft SQL Server DW, Netezza, Oracle DW, Teradata, or Vertica.</td>
</tr>
</tbody>
</table>

4. Choose **OK** to create your AWS SCT project.
Converting Your Schema

Use the following procedure to convert schema.

To convert schema

1. Choose View, and then choose Main View.

2. In the left panel that displays the schema from your source database, choose a schema object to convert. Open the context (right-click) menu for the object, and then choose Convert schema.

3. When AWS SCT finishes converting the schema, you can view the proposed schema in the panel on the right of your project.

At this point, no schema is applied to your target Amazon RDS DB instance. The planned schema is part of your project. If you select a converted schema item, you can see the planned schema command in the panel at lower center for your target Amazon RDS DB instance.
You can edit the schema in this window. The edited schema is stored as part of your project and is written to the target DB instance when you choose to apply your converted schema.

Applying the Converted Schema to Your Target DB Instance

You can apply the converted database schema to your target Amazon RDS DB instance. After the schema has been applied to your target DB instance, you can update the schema based on the action items in the database migration assessment report.

Warning
This procedure overwrites the existing target schema. Be careful not to overwrite schema unintentionally. Be careful not to overwrite schema in your target DB instance that you have already modified, or you will overwrite those changes.

To apply the converted database schema to your target Amazon RDS DB instance

1. Choose the schema element in the right panel of your project that displays the planned schema for your target DB instance.
2. Open the context (right-click) menu for the schema element, and then choose Apply to database.
The converted schema is applied to the target DB instance.
Getting Started with the AWS Schema Conversion Tool

You can use the AWS Schema Conversion Tool (AWS SCT) to convert your source database schema to a schema for databases hosted on Amazon Web Services (AWS). The AWS SCT application provides a project-based user interface. Almost all work you do with AWS SCT starts with the following steps:

1. Install AWS SCT. For more information, see Installing, Verifying, and Updating the AWS Schema Conversion Tool (AWS SCT) (p. 4).
2. Install a data extractor agent if you want to migrate data from a data warehouse to Amazon Redshift. For more information, see Using Data Extraction Agents (p. 110).
3. Familiarize yourself with the user interface of AWS SCT. For more information, see Using the AWS Schema Conversion Tool (AWS SCT) User Interface (p. 12).
4. Create an AWS SCT project. Connect to your source and target databases. For more information about connecting to your source database, see Source Databases for the AWS Schema Conversion Tool (p. 29).
5. Run and then review the Database Migration Assessment Report. For more information about the assessment report, see Creating and Reviewing the Database Migration Assessment Report (p. 20).
6. Convert the source database schemas. There are several aspects of the conversion you need to keep in mind, such as what to do with items that don't convert, and how to map items that should be converted a particular way. For more information about converting a source schema, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 70).

If you are converting a data warehouse schema, there are also aspects you need to consider before doing the conversion. For more information, see Converting Data Warehouse Schemas to Amazon Redshift by Using the AWS Schema Conversion Tool (p. 88).
7. Applying the schema conversion to your target. For more information about applying a source schema conversion, see Using the AWS Schema Conversion Tool (AWS SCT) User Interface (p. 12).
8. The AWS SCT can also be used to convert SQL stored procedures and other application code. For more information, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 134)

You can also use AWS SCT to migrate your data from a source database to an Amazon-managed database. For more information, see
Source Databases for the AWS Schema Conversion Tool

AWS Schema Conversion Tool (AWS SCT) can convert the following source database schemas to a target database. Select the link below for information on permissions required, connection information, and information on what AWS SCT can convert for use with the target database.

Topics
- Using Oracle as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 29)
- Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 37)
- Using MySQL as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 45)
- Using PostgreSQL as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 47)
- Using Db2 LUW as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 49)
- Using Amazon Redshift as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 54)
- Using Oracle DW as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 56)
- Using Teradata as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 58)
- Using Netezza as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 61)
- Using Greenplum as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 63)
- Using Vertica as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 65)
- Using Microsoft SQL Server DW as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 67)

Using Oracle as a Source for AWS Schema Conversion Tool (AWS SCT)

Topics
- Permissions Required When Using Oracle as a Source Database (p. 30)
- Connecting to Oracle as a Source Database (p. 30)
- Converting an Oracle Database to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL) (p. 33)
- Converting an Oracle Database to Amazon RDS for MySQL or Amazon Aurora (MySQL) (p. 35)
- Converting Oracle to Amazon RDS for Oracle (p. 36)

You can use AWS SCT to convert data from Oracle to the following targets:
- Amazon RDS for MySQL
- Amazon Aurora (MySQL)
- Amazon RDS for PostgreSQL
Permissions Needed for Oracle as a Source

- Amazon Aurora (PostgreSQL)
- Amazon RDS for Oracle

When the source is an Oracle database, comments can be converted to the appropriate format in, for example, a PostgreSQL database. AWS SCT can convert comments on tables, views, and columns. Comments can include apostrophes; AWS SCT doubles the apostrophes when converting SQL statements, just as it does for string literals.

For Oracle to Amazon RDS for Oracle conversions, DB Links is supported. A database link is a schema object in one database that enables you to access objects on another database. The other database doesn't need to be an Oracle database. However, to access non-Oracle databases you must use Oracle Heterogeneous Services.

Once you create a database link, you can use the link in SQL statements to refer to tables, views, and PL/SQL objects in the other database. To use a database link, append @dblink to the table, view, or PL/SQL object name. You can query a table or view in the other database with the SELECT statement. For more information about using Oracle database links, see the Oracle documentation.

For more information about using database links with Amazon RDS, see the Amazon RDS documentation.

Permissions Required When Using Oracle as a Source Database

The privileges required for Oracle as a source are listed following:

- CONNECT
- SELECT_CATALOG_ROLE
- SELECT ANY DICTIONARY

Connecting to Oracle as a Source Database

Use the following procedure to connect to your Oracle source database with the AWS Schema Conversion Tool (AWS SCT).

To connect to an Oracle source database

1. In the AWS Schema Conversion Tool, choose Connect to Oracle.

The Connect to Oracle dialog box appears.
2. Provide the Oracle source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Choose the connection type to your database. Depending on your type, provide the following additional information:</td>
</tr>
<tr>
<td></td>
<td>• <strong>SID</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>Server name</strong>: The DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Server port</strong>: The port used to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Oracle SID</strong>: The Oracle System ID (SID). To find the Oracle SID, submit the following query to your Oracle database:</td>
</tr>
<tr>
<td></td>
<td>SELECT sys_context('userenv','instance_name') AS SID FROM dual;</td>
</tr>
<tr>
<td></td>
<td>• <strong>Service Name</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>Server name</strong>: The DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Server port</strong>: The port used to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Service Name</strong>: The name of the Oracle service to connect to.</td>
</tr>
<tr>
<td></td>
<td>• <strong>TNS Alias</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>TNS file path</strong>: The path to the file that contains the Transparent Network Substrate (TNS) name connection information.</td>
</tr>
<tr>
<td></td>
<td>• <strong>TNS file path</strong>: The TNS alias from this file to use to connect to the source database.</td>
</tr>
<tr>
<td></td>
<td>• <strong>TNS Connect Identifier</strong></td>
</tr>
</tbody>
</table>
## Connecting to Oracle as a Source

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TNS identifier</strong></td>
<td>The identifier for the registered TNS connection information.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Type the user name and password to connect to your source database server. The first time you connection to the Oracle database, you have to enter the path to the Oracle Driver file (ojdbc7.jar). You can download the file at <a href="http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html">http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html</a>. You must register on the free Oracle Technical Network website to complete the download. AWS SCT uses the selected driver for any future Oracle database connections. The driver path can be modified using the Drivers tab in Global Settings. <strong>Note</strong> AWS SCT uses the password to connect to your source database only when you create your project or choose the Connect to source option in a project, where source is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:</td>
</tr>
<tr>
<td><strong>SSL Authentication</strong></td>
<td>Select this option to use SSL authentication by certificate is used for the connection instead of user name and password.</td>
</tr>
<tr>
<td><strong>Trust Store</strong></td>
<td>The location of a trust store containing certificates.</td>
</tr>
<tr>
<td><strong>Trust Store Password</strong></td>
<td>The password for the trust store.</td>
</tr>
<tr>
<td><strong>Key Store</strong></td>
<td>The location of a key store containing a private key and certificates. This value is required if SSL Authentication is selected and is otherwise optional.</td>
</tr>
<tr>
<td><strong>Trust Store Password</strong></td>
<td>The password for the key store. This value is required if SSL Authentication is selected and is otherwise optional.</td>
</tr>
<tr>
<td><strong>Store Password</strong></td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td><strong>Oracle Driver Path</strong></td>
<td>Type the path to the driver to use to connect to the source database. For more information, see <a href="#">Installing the Required Database Drivers</a> (p. 8). If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see <a href="#">Storing Driver Paths in the Global Settings</a> (p. 10).</td>
</tr>
</tbody>
</table>
Converting an Oracle Database to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL)

AWS SCT can convert SQL*Plus files into PSQL. The assessment report shows how AWS SCT converted the SQL*Plus files into PSQL. To convert SQL*Plus files to PSQL, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 134).

This section covers the following topics:

**Topics**
- Converting Dynamic SQL for Oracle to PostgreSQL Migrations (p. 33)
- Converting Oracle Partitions to PostgreSQL Version 10 Partitions (p. 34)

When converting Oracle system objects to PostgreSQL, AWS SCT converts the following:

<table>
<thead>
<tr>
<th>Oracle system object</th>
<th>Description</th>
<th>Converted PostgreSQL object</th>
</tr>
</thead>
<tbody>
<tr>
<td>V$VERSION</td>
<td>Displays version numbers of core library components in the Oracle Database</td>
<td>aws_oracle_ext.v $version</td>
</tr>
<tr>
<td>V$INSTANCE</td>
<td>A view that shows the state of the current instance.</td>
<td>aws_oracle_ext.v $instance</td>
</tr>
</tbody>
</table>

After AWS SCT migrates schema objects and stored data, you can redirect your workflow from the source database to the target. In most cases, work flows use unique numbers generated by database sequences to maintain integrity constraints such as primary keys and unique keys for data written to the target database. This means that new values of a migrated sequence and the values generated before the migration must not overlap.

For Oracle to PostgreSQL migration projects, you can make sure that there is no overlap. To do so, choose the option *Populate converted sequences with the last values generated on the source side* in the Conversion settings tab of Project Settings.

**Converting Dynamic SQL for Oracle to PostgreSQL Migrations**

Dynamic SQL is a programming technique that you can use to run data definition language (DDL) statements inside PL/SQL code. You can also use dynamic SQL to generate and run SQL statements at run time when you don't know the exact text or object identifiers during development. AWS SCT can convert dynamic SQL statements used with Oracle databases to their analog statements in PostgreSQL.

**To convert Oracle dynamic SQL to PostgreSQL SQL**

1. Create an Oracle-to-PostgreSQL migration project.
2. Connect to the source and target databases.
3. Choose a stored procedure in the Oracle source tree view. The procedure should contain references to the DBMS_SQL Oracle package or have an EXECUTE IMMEDIATE statement.
4. For **Actions**, choose **Convert Schema**, and agree to replace the objects if they exist. The following screenshot shows the converted procedure below the Oracle procedure.

---

**Converting Oracle Partitions to PostgreSQL Version 10**

**Partitions**

In PostgreSQL version 10 and later, you can specify how to divide a table into parts called *partitions*. The table that is divided is called a *partitioned table*. The table’s specification includes the partitioning method and a list of columns or expressions used as the partition key.

All rows inserted into a partitioned table are routed to one of the partitions based on the value of the partition key. Each partition has a subset of the data defined by its partition bounds. Currently supported partitioning methods include range and list. In range partitioning, each partition is assigned a range of keys. In list partitioning, each partition is assigned a list of keys.

AWS SCT can emulate partitions and subpartitions when converting a schema from an Oracle database to a PostgreSQL database. An Oracle subpartition is converted to a PostgreSQL partition that has a table as its parent range expression. The table is partitioned by range expression from the original Oracle subpartition.

AWS SCT currently supports the following partitioning scenarios:

- Range
- List
• Range-Range
• List-List
• Range-List
• List-Range

The following scenarios are not currently supported:
• Hash
• Range-Hash
• List-Hash
• Interval
• Reference
• System

Some known issues with partition conversion to PostgreSQL version 10 include the following:
• Only not-null columns can be partitioned by columns.
• DEFAULT is not a possible value for a partition value.
• Partitions based on TIMESTAMP are not supported.
• Partitions based on a hash function are not supported.
• You can’t update columns in partitioned tables because updating a value might cause the value to go to a different partition. AWS SCT only supports deleting and inserting columns in partitioned tables.
• Foreign keys are not supported going from and to partitioned tables.

Converting an Oracle Database to Amazon RDS for MySQL or Amazon Aurora (MySQL)

Some things to consider when migrating a SQL Server schema to ToPostgreSQL:
• A GOTO statement and a label can be used to change the order that statements are run in. Any Transact-SQL statements that follow a GOTO statement are skipped and processing continues at the label. GOTO statements and labels can be used anywhere within a procedure, batch, or statement block. GOTO statements can also be nested.

MySQL doesn’t use GOTO statements. When AWS SCT converts code that contains a GOTO statement, it converts the statement to use a BEGIN…END or LOOP…END LOOP statement. You can find examples of how AWS SCT converts GOTO statements in the table following.

<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>BEGIN</td>
</tr>
<tr>
<td>....</td>
<td>labell1:</td>
</tr>
<tr>
<td>statement1;</td>
<td>BEGIN</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>GOTO labell1;</td>
<td>statement1;</td>
</tr>
<tr>
<td>statement2;</td>
<td>....</td>
</tr>
<tr>
<td>....</td>
<td>LEAVE labell1;</td>
</tr>
<tr>
<td>labell1:</td>
<td>statement2;</td>
</tr>
<tr>
<td>Statement3;</td>
<td>....</td>
</tr>
</tbody>
</table>
### Converting Oracle to Amazon RDS for Oracle

Some things to consider when migrating Oracle schema and code to Amazon RDS for Oracle:

- **AWS SCT can add directory objects to the object tree.** Directory objects are logical structures that each represent a physical directory on the server's file system. You can use directory objects with packages such as DBMS_LOB, UTL_FILE, DBMS_FILE_TRANSFER, the DATAPUMP utility, and so on.

- **AWS SCT supports converting Oracle tablespaces to an Amazon RDS for Oracle DB instance.** Oracle stores data logically in tablespaces and physically in data files associated with the corresponding tablespace. In Oracle you can create tablespace with data file names. Amazon RDS supports Oracle Managed Files (OMF) for data files, log files and control files only. AWS SCT creates the needed data files during conversion.

- **AWS SCT can convert server-level roles and privileges.** The Oracle database engine uses role-based security. A role is a collection of privileges that you can grant to or revoke from a user. A predefined role in Amazon RDS, called DBA, normally allows all administrative privileges on an Oracle database engine. The following privileges are not available for the DBA role on an Amazon RDS DB instance using the Oracle engine:
  - Alter database
• Alter system
• Create any directory
• Grant any privilege
• Grant any role
• Create external job

You can grant all other privileges to an Oracle RDS user role.

Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert schemas and application code from SQL Server to the following targets:
• Amazon RDS for MySQL
• Amazon Aurora (MySQL)
• Amazon RDS for PostgreSQL
• Amazon Aurora (PostgreSQL)
• Amazon RDS for SQL Server

These are a few things to consider when using SQL Server as a source for AWS SCT:
• Amazon RDS has limited support for linked servers. When converting SQL Server application code that uses linked servers, AWS SCT converts the application code but you should review the behavior of objects that use link servers before you run the converted code.
• PATINDEX returns the starting position of the first occurrence of a pattern in a specified expression, or zeros if the pattern is not found, on all valid text and character data types. When converting from SQL Server to Amazon RDS for PostgreSQL, AWS SCT replaces application code that uses PATINDEX with aws_sqlserver_ext.patindex(<pattern character>, <expression character varying>) .

When converting from SQL Server to PostgreSQL, AWS SCT converts SQL Server system objects into recognizable objects in PostgreSQL. The following table shows how the system objects are converted.

<table>
<thead>
<tr>
<th>MS SQL Server Use Cases</th>
<th>PostgreSQL Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS.SCHEMAS</td>
<td>AWS_SQLSERVER_EXT.SYS_SCHEMAS</td>
</tr>
<tr>
<td>SYS.TABLES</td>
<td>AWS_SQLSERVER_EXT.SYS_TABLES</td>
</tr>
<tr>
<td>SYS.VIEWS</td>
<td>AWS_SQLSERVER_EXT.SYS_VIEWS</td>
</tr>
<tr>
<td>SYS.ALL_VIEWS</td>
<td>AWS_SQLSERVER_EXT.SYS_ALL_VIEWS</td>
</tr>
<tr>
<td>SYS.TYPES</td>
<td>AWS_SQLSERVER_EXT.SYS_TYPES</td>
</tr>
<tr>
<td>SYS.COLUMNS</td>
<td>AWS_SQLSERVER_EXT.SYS_COLUMNS</td>
</tr>
<tr>
<td>SYS.ALL_COLUMNS</td>
<td>AWS_SQLSERVER_EXT.SYS_ALL_COLUMNS</td>
</tr>
<tr>
<td>SYS.FOREIGN_KEYS</td>
<td>AWS_SQLSERVER_EXT.SYS_FOREIGN_KEYS</td>
</tr>
<tr>
<td>SYS.SYSFOREIGNKEYS</td>
<td>AWS_SQLSERVER_EXT.SYS_SYSFOREIGNKEYS</td>
</tr>
</tbody>
</table>
## Permissions Required When Using Microsoft SQL Server as a Source

The privileges required for Microsoft SQL Server as a source are listed following:

- VIEW DEFINITION
- VIEW DATABASE STATE

Repeat the grant for each database whose schema you are converting.
Using Windows Authentication When Using Microsoft SQL Server as a Source

If your application runs on a Windows-based intranet, you might be able to use Windows Authentication for database access. Windows Authentication uses the current Windows identity established on the operating system thread to access the SQL Server database. You can then map the Windows identity to a SQL Server database and permissions. To connect to SQL Server using Windows Authentication, you must specify the Windows identity that your application is using. You must also grant the Windows identity access to the SQL Server database.


The possible example for creating a user in TEST_DB is shown below

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

Using Windows Authentication with a JDBC Connection

The JDBC driver does not support Windows Authentication when the driver is used on non-Windows operating systems. Windows Authentication credentials, such as user name and password, when connecting to SQL Server from non-Windows operating systems. In such cases, the applications must use SQL Server Authentication instead.

In JDBC connection string, the parameter integratedSecurity must be specified to connect using Windows Authentication. The JDBC driver supports Integrated Windows Authentication on Windows operating systems through the integratedSecurity connection string parameter.

To use integrated authentication

1. Install the JDBC driver.
2. Copy the sqljdbc_auth.dll file to a directory on the Windows system path on the computer where the JDBC driver is installed.

The sqljdbc_auth.dll files are installed in the following location:

\<installation directory>\sqljdbc_<version>\<language>\auth\x86

When you try to establish a connection to SQL Server database using Windows Authentication, you might get the error: This driver is not configured for integrated authentication. This problem can be solved by performing the following actions:

- need to declare two variables which point to the installed path of your JDBC:
  - variable name: SQLJDBC_HOME; variable value: D:\lib\JDBC4.1\enu (where your sqljdbc4.jar exists);
  - variable name: SQLJDBC_AUTH_HOME; variable value: D\lib\JDBC4.1\enu\auth\x86 (if you are running 32bit OS) or D\lib\JDBC4.1\enu\auth\x64 (if you are running 64bit OS). This is where your sqljdbc_auth.dll is located.
• copy sqljdbc_auth.dll to folder where your JDK/JRE is running. You may copy to lib folder, bin folder, etc. I copied to the following folder:

[\JDK_INSTALLED_PATH]\bin;
[\JDK_INSTALLED_PATH]\jre\bin;
[\JDK_INSTALLED_PATH]\jre\lib;
[\JDK_INSTALLED_PATH]\lib;

• ensure that in your jdbc library folder, you only have SQLJDBC4.jar. Please remove other sqljdbc*.jar file from that folder (or copy to other folder). If you are adding the driver as part of your program, please ensure that you add only SQLJDBC4.jar as driver to use.

• copy sqljdbc_auth.dll the file in the folder with your application.

**Note**
If you are running a 32-bit Java Virtual Machine (JVM), use the sqljdbc_auth.dll file in the x86 folder, even if the operating system is the x64 version. If you are running a 64-bit JVM on a x64 processor, use the sqljdbc_auth.dll file in the x64 folder.

When you connect to a SQL Server database, you can choose either the **Windows Authentication** or **SQL Server Authentication** for the **Authentication** option.

**Connecting to SQL Server as a Source**

Use the following procedure to connect to your Microsoft SQL Server source database with the AWS Schema Conversion Tool (AWS SCT).

**To connect to a Microsoft SQL Server source database**

1. In the AWS Schema Conversion Tool, choose **Connect to Microsoft SQL Server**.

The **Connect to Microsoft SQL Server** dialog box appears.
2. Provide the Microsoft SQL Server source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the Domain Name Service (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Instance name</td>
<td>Type the instance name for the SQL Server database. To find the instance name, run the query SELECT @@servername; on your SQL Server database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server. Note AWS SCT uses the password to connect to your source database only when you create your project or choose the Connect to source option in a project, where source is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Select this option if you want to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:</td>
</tr>
<tr>
<td></td>
<td>• Trust Server Certificate: Select this option to trust the server certificate.</td>
</tr>
<tr>
<td></td>
<td>• Trust Store: The location of a trust store containing certificates.</td>
</tr>
</tbody>
</table>
For This Parameter | Do This
---|---
| • **Trust Store Password**: The password for the trust store.

**Store Password** | AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.

**Sql Server Driver Path** | Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8).

If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10).

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.

### Converting a SQL Server Schema to MySQL

Some things to consider when migrating a SQL Server schema to MySQL:

- MySQL doesn’t support the MERGE statement. However, AWS SCT can emulate the MERGE statement during conversion by using the INSERT ON DUPLICATE KEY clause and the UPDATE FROM and DELETE FROM statements.

  For correct emulation using INSERT ON DUPLICATE KEY, make sure that a unique constraint or primary key exists on the target MySQL database.

- A GOTO statement and a label can be used to change the order that statements are run in. Any Transact-SQL statements that follow a GOTO statement are skipped and processing continues at the label. GOTO statements and labels can be used anywhere within a procedure, batch, or statement block. GOTO statements can also be nested.

MySQL doesn’t use GOTO statements. When AWS SCT converts code that contains a GOTO statement, it converts the statement to use a BEGIN…END or LOOP…END LOOP statement. You can find examples of how AWS SCT converts GOTO statements in the table following.

<table>
<thead>
<tr>
<th>SQL Server statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN .... statement1; .... GOTO label1; statement2; .... label1: Statement3; END</td>
<td>BEGIN label1: BEGIN .... statement1; .... LEAVE label1; statement2; .... END; Statement3; END</td>
</tr>
</tbody>
</table>
### Converting a SQL Server to PostgreSQL

Some things to consider when migrating a SQL Server schema to PostgreSQL:

- In PostgreSQL, all object's names in a schema must be unique, including indexes. Index names must be unique in the schema of the base table. In SQL Server, an index name can be the same for different tables.

To ensure the uniqueness of index names, AWS SCT gives you the option to generate unique index names if your index names are not unique. To do this, choose the option **Generate unique index names** in the project properties. By default, this option is enabled. If this option is enabled, unique index names are created using the format `IX_table_name_index_name`. If this option is disabled, index names aren’t changed.

- A GOTO statement and a label can be used to change the order that statements are run in. Any Transact-SQL statements that follow a GOTO statement are skipped and processing continues at the label. GOTO statements and labels can be used anywhere within a procedure, batch, or statement block. GOTO statements can also be nested.

PostgreSQL doesn’t use GOTO statements. When AWS SCT converts code that contains a GOTO statement, it converts the statement to use a `BEGIN...END` or `LOOP...END LOOP` statement. You can find examples of how AWS SCT converts GOTO statements in the table following.

<table>
<thead>
<tr>
<th>SQL Server statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>BEGIN</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>statement1;</td>
<td>statement1;</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>label1:</td>
<td>label1:</td>
</tr>
<tr>
<td>statement2;</td>
<td>LOOP</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>GOTO label1;</td>
<td>ITERATE label1;</td>
</tr>
<tr>
<td>statement3;</td>
<td>LEAVE label1;</td>
</tr>
<tr>
<td>....</td>
<td>END LOOP;</td>
</tr>
<tr>
<td>statement4;</td>
<td>statement3;</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SQL Server statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td>BEGIN</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>statement1;</td>
<td>statement1;</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>label1:</td>
<td>label1:</td>
</tr>
<tr>
<td>statement2;</td>
<td>BEGIN</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>statement3;</td>
<td>statement2;</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>statement4;</td>
<td>statement3;</td>
</tr>
<tr>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>END</td>
<td>END;</td>
</tr>
</tbody>
</table>

<p>| Version 1.0 | 43 |</p>
<table>
<thead>
<tr>
<th>SQL Server statement</th>
<th>PostgreSQL statement</th>
</tr>
</thead>
</table>
| BEGIN
| ...                  |
| statement1;          |
| ...                  |
| GOTO label1;         |
| statement2;          |
| ...                  |
| label1:              |
| Statement3;          |
| ...                  |
| END                  | BEGIN
| label1:              |
| BEGIN                |
| ...                  |
| statement1;          |
| ...                  |
| EXIT label1;         |
| statement2;          |
| ...                  |
| END;                 |
| Statement3;          |
| ...                  |
| END                  |
| BEGIN                |
| ...                  |
| statement1;          |
| ...                  |
| label1:              |
| statement2;          |
| ...                  |
| label1:              |
| statement3;          |
| ...                  |
| statement4;          |
| ...                  |
| END                  | BEGIN
| ...                  |
| statement1;          |
| ...                  |
| label1:              |
| LOOP                 |
| statement2;          |
| ...                  |
| CONTINUE label1;      |
| EXIT label1;         |
| END LOOP;            |
| statement3;          |
| ...                  |
| statement4;          |
| ...                  |
| END                  |
| BEGIN                |
| ...                  |
| statement1;          |
| ...                  |
| label1:              |
| statement2;          |
| ...                  |
| label1:              |
| BEGIN                |
| statement2;          |
| ...                  |
| statement3;          |
| ...                  |
| statement4;          |
| ...                  |
| END;                 |

- AWS SCT can add database triggers to the object tree when Amazon RDS is the target.
- AWS SCT can add server-level triggers to the object tree when Amazon RDS is the target.
- AWS SCT can add linked servers to the object tree when Amazon RDS is the target.
Converting SQL Server Partitions to PostgreSQL Version 10 Partitions

In SQL Server, you create partitions with partition functions. When converting from a SQL Server portioned table to a PostgreSQL version 10 partitioned table, be aware of several potential issues:

- SQL Server allows you to partition a table using a column without a NOT NULL constraint. In that case, all NULL values go to the leftmost partition. PostgreSQL doesn't support NULL values for RANGE partitioning.
- SQL Server allows you to create primary and unique keys for partitioned tables. For PostgreSQL, you create primary or unique keys for each partition directly. Thus, PRIMARY or UNIQUE KEY constraint must be removed from their parent table when migrating to PostgreSQL. The resulting key names take the format 
  `<original_key_name>_<partition_number>`.
- SQL Server allows you to create foreign key constraint from and to partitioned tables. PostgreSQL doesn't support foreign keys referencing partitioned tables. Also, PostgreSQL doesn't support foreign key references from a partitioned table to another table.
- SQL Server allows you to create indexes for partitioned tables. For PostgreSQL, an index should be created for each partition directly. Thus, indexes must be removed from their parent tables when migrating to PostgreSQL. The resulting index names take the format
  `<original_index_name>_<partition_number>`.
- PostgreSQL doesn't support partitioned indexes.

Converting SQL Server to Amazon RDS for SQL Server

Some things to consider when migrating SQL Server schema and code to Amazon RDS for SQL Server:

- AWS SCT can convert SQL Server Agent to provide schedules, alerts, and jobs on an Amazon RDS for SQL Server DB instance. After conversion, you can use an Amazon RDS for SQL Server DB instance as a data source for SQL Server Reporting Service (SSRS), SQL Server Analysis Services (SSAS), and SQL Server Integration Services (SSIS). You can't run these services on the DB instance.

Using MySQL as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert schemas and application code from MySQL to the following targets:

- Amazon RDS for PostgreSQL
- Amazon Aurora (PostgreSQL)
- Amazon RDS for MySQL
- Amazon Aurora (MySQL)

Privileges for MySQL as a Source Database

The privileges required for MySQL as a source are listed following:
Connecting to MySQL as a Source Database

Use the following procedure to connect to your MySQL source database with the AWS Schema Conversion Tool (AWS SCT).

To connect to a MySQL source database

1. In the AWS Schema Conversion Tool, choose Connect to MySQL.

The Connect to MySQL dialog box appears.

2. Provide the MySQL source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
</tbody>
</table>

Note
AWS SCT uses the password to connect to your source database only when you create your project or choose the Connect to source option in a project, where source is your source database. To guard against exposing the password for your source database, AWS...
<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SCT doesn’t store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
</tbody>
</table>

**Use SSL**

Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:

- **Require SSL**: Select this option if you want to connect to the server only through SSL.
  
  **Note**
  
  If you choose Require SSL, it means that if the server doesn’t support SSL, you can’t connect to the server. If you don’t choose Require SSL and the server doesn’t support SSL, you can still connect to the server without using SSL. For more information, see Using Secure Connections.

- **Verify Server Certificate**: Select this option to verify the server certificate by using a trust store.

- **Trust Store**: The location of a trust store containing certificates.

- **Trust Store Password**: The password for the trust store.

**Store Password**

AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.

**MySql Driver Path**

Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8).

If you store the driver path in the global project settings, the driver path doesn’t appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10).

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.

---

**Using PostgreSQL as a Source for AWS Schema Conversion Tool (AWS SCT)**

You can use AWS SCT to convert data from PostgreSQL to the following targets:

- connecting to source, connecting to target, reference info, permissions

- Amazon RDS for MySQL
- Amazon Aurora (MySQL)
- Amazon RDS for PostgreSQL
- Amazon Aurora (PostgreSQL)
Privileges for PostgreSQL as a Source Database

The privileges required for PostgreSQL as a source are listed following:

- 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>`

Connecting to PostgreSQL as a Source

Use the following procedure to connect to your PostgreSQL source database with the AWS Schema Conversion Tool (AWS SCT).

To connect to a PostgreSQL source database

1. In the AWS Schema Conversion Tool, choose Connect to PostgreSQL.

   ![Connect to PostgreSQL dialog box](image)

   The Connect to PostgreSQL dialog box appears.

2. Provide the PostgreSQL source database connection information. Use the instructions in the following table.
<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Type the name of the PostgreSQL database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>Note: AWS SCT uses the password to connect to your source database only when you create your project or choose the Connect to source option in a project, where source is your source database. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:</td>
</tr>
<tr>
<td></td>
<td>• Verify Server Certificate: Select this option to verify the server certificate by using a trust store.</td>
</tr>
<tr>
<td></td>
<td>• Trust Store: The location of a trust store containing certificates.</td>
</tr>
<tr>
<td></td>
<td>• Trust Store Password: The password for the trust store.</td>
</tr>
<tr>
<td>Store Password</td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td>PostgreSQL Driver Path</td>
<td>Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8).</td>
</tr>
<tr>
<td></td>
<td>If you store the driver path in the global project settings, the driver path doesn’t appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10).</td>
</tr>
</tbody>
</table>

3. Choose Test Connection to verify that you can successfully connect to your source database.
4. Choose OK to connect to your source database.

Using Db2 LUW as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert data from Db2 LUW to the following targets. AWS SCT supports as a source Db2 LUW versions 9.1, 9.5, 9.7, 10.1, 10.5, and 11.1.

- Amazon RDS for MySQL
Permissions Needed When Using Db2 LUW as a Source

The privileges needed to connect to a DB2LUW database, to check available privileges and read schema metadata for a source are listed following:

- Privilege needed to establish a connection:
  ```sql
  GRANT CONNECT ON DATABASE TO USER min_privs;
  ```
- Privilege needed to run SQL statements:
  ```sql
  GRANT EXECUTE ON PACKAGE NULLID.SYSSH200 TO USER MIN_PRIVS;
  ```
- Privileges needed to get instance-level information:
  ```sql
  GRANT EXECUTE ON FUNCTION SYSPROC.ENV_GET_INST_INFO TO USER MIN_PRIVS;
  GRANT SELECT ON SYSBMADM.ENV_INST_INFO TO USER MIN_PRIVS;
  GRANT SELECT ON SYSBMADM.ENV_SYS_INFO TO USER MIN_PRIVS;
  ```
- Privileges needed to check privileges granted through roles, groups and authorities:
  ```sql
  GRANT EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID TO USER MIN_PRIVS;
  GRANT EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID TO USER MIN_PRIVS;
  GRANT EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID TO USER MIN_PRIVS;
  GRANT SELECT ON SYSBMADM.PRIVILEGES TO USER MIN_PRIVS;
  ```
- Privileges needed on system catalogs and tables:
  ```sql
  GRANT SELECT ON SYSCAT.ATTRIBUTES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.CHECKS TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.COLIDENTATTRIBUTES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.COLUMNS TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.DATAPARTITIONEXPRESSION TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.DATAPARTITIONS TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.DATATYPEDEP TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.DATATYPES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.HIERARCHIES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.INDEXCOLUSE TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.INDEXES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.INDEXPARTITIONS TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.KEYCOLUSE TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.MODULEOBJECTS TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.MODULES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.NICKNAMES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.PERIODS TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.REFERENCES TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.ROUTINEPARMS TO USER MIN_PRIVS;
  GRANT SELECT ON SYSCAT.ROUTINES TO USER MIN_PRIVS;
  ```
• GRANT SELECT ON SYSCAT.ROWFIELDS TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.SCHEMATA TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.SEQUENCES TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.TABCONST TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.TABLES TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.TRIGGERS TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.VARIABLEDEP TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.VARIABLES TO USER MIN_PRIVS;
• GRANT SELECT ON SYSCAT.VIEWS TO USER MIN_PRIVS;
• GRANT SELECT ON SYSIBM.SYSDUMMY1 TO USER MIN_PRIVS;

• To run SQL statements, the user account needs a privilege to use at least one of the workloads enabled in the database. If none of the workloads are assigned to the user, ensure that the default user workload is accessible to the user:

GRANT USAGE ON WORKLOAD SYSDEFAULTUSERWORKLOAD TO USER MIN_PRIVS;

To execute queries, you need to create system temporary tablespace with page size 8K, 16K and 32K, if they don't exist. To create the temporary tablespaces, run the following scripts:

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;

Connecting to a Db2 LUW Source

Use the following procedure to connect to your Db2 LUW source database with the AWS Schema Conversion Tool (AWS SCT).
To connect to a Db2 LUW source database

1. In the AWS Schema Conversion Tool, choose **Connect to Source DB2 LUW**.

   ![Connect to DB2 LUW dialog box]

   The **Connect to DB2 LUW** dialog box appears.

2. Provide the Db2 LUW source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server name</strong></td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Type the name of the Db2 LUW database.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you create your project or choose the <strong>Connect to source</strong> option in a project, where <strong>source</strong> is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the <strong>SSL</strong> tab:</td>
</tr>
</tbody>
</table>
For This Parameter | Do This
--- | ---
• **Verify Server Certificate**: Select this option to verify the server certificate by using a trust store.
• **Trust Store**: The location of a trust store containing certificates. You must add this location in Global Settings before it will appear here.
• **Trust Store Password**: The password for the trust store.

For more information about SSL support for Db2 LUW, see Configure Security Options for Connections.

<table>
<thead>
<tr>
<th>Store Password</th>
<th>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DB2 LUW Driver Path</th>
<th>Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8). If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10).</th>
</tr>
</thead>
</table>

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.

**Converting an DB2 LUW Database to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL)**

Some things to consider when migrating IBM Db2 LUW to ToPostgreSQL:

• **AWS SCT** can convert various trigger statements used with Db2 LUW. These trigger statements include the following:
  • Trigger events - INSERT, DELETE, and UPDATE trigger events specify that the triggered action runs whenever the event is applied to the subject table or subject view. You can specify any combination of the INSERT, DELETE, and UPDATE events, but you can specify each event only once. AWS SCT supports single and multiple trigger events. For events, PostgreSQL has practically the same functionality.
  • Event OF COLUMN - You can specify a column name from a base table. The trigger is activated only by the update of a column that is identified in the column-name list. PostgreSQL has the same functionality.
  • Statement triggers – These specify that the triggered action is applied only once for the whole statement. You can’t specify this type of trigger granularity for a BEFORE trigger or an INSTEAD OF trigger. If specified, an UPDATE or DELETE trigger is activated, even if no rows are affected. PostgreSQL also has this functionality and trigger declaration for statement triggers is identical for PostgreSQL and Db2 LUW.
  • Referencing clauses – These specify the correlation names for transition variables and the table names for transition tables. Correlation names identify a specific row in the set of rows affected by the triggering SQL operation. Table names identify the complete set of affected rows. Each row affected by a triggering SQL operation is available to the triggered action by qualifying columns
with specified correlation-names. PostgreSQL doesn't support this functionality, and only uses a NEW or OLD correlation name.

- AWS SCT supports INSTEAD OF triggers.

Using Amazon Redshift as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert data from Amazon Redshift to the following targets:

- Amazon Redshift

Privileges for Amazon Redshift as a Source Database

The privileges required for using Amazon Redshift as a source are listed following:

- 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

Connecting to Redshift as a Source

Use the following procedure to connect to your Amazon Redshift source database with the AWS Schema Conversion Tool (AWS SCT).

To connect to an Amazon Redshift source database

1. In the AWS Schema Conversion Tool, choose Connect to Source Amazon Redshift.

The Connect to Amazon Redshift dialog box appears.
2. Provide the Amazon Redshift source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Type the name of the Amazon Redshift database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
</tbody>
</table>

**Note**
AWS SCT uses the password to connect to your source database only when you create your project or choose the **Connect to source** option in a project, where *source* is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.

<table>
<thead>
<tr>
<th>Use SSL</th>
<th>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the <strong>SSL</strong> tab:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Verify Server Certificate</strong>: Select this option to verify the server certificate by using a trust store.</td>
</tr>
<tr>
<td></td>
<td><strong>Trust Store</strong>: The location of a trust store containing certificates. You must add this location in Global Settings before it will appear here.</td>
</tr>
<tr>
<td></td>
<td><strong>Trust Store Password</strong>: The password for the trust store.</td>
</tr>
</tbody>
</table>
For This Parameter | Do This
--- | ---
For more information about SSL support for Amazon Redshift, see [Configure Security Options for Connections](#).
Store Password | AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.
Redshift Driver Path | Type the path to the driver to use to connect to the source database. For more information, see [Installing the Required Database Drivers (p. 8)](#).
If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see [Storing Driver Paths in the Global Settings (p. 10)](#).

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.

### Using Oracle DW as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert data from Oracle DW to Amazon Redshift.

#### Privileges for Oracle Data Warehouse as a Source

The privileges required for Oracle Data Warehouse as a source are listed following:

- `connect`
- `select_catalog_role`
- `select any dictionary`

#### Connecting to OracleDW as a Source

Use the following procedure to connect to your Oracle data warehouse source database with the AWS Schema Conversion Tool (AWS SCT).

**To connect to an Oracle data warehouse source database**

1. In the AWS Schema Conversion Tool, choose **Connect to Oracle DW**.

The **Connect to Oracle** dialog box appears.
2. Provide the Oracle Data Warehouse source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Choose the connection type to your database. Depending on your type, provide the following additional information:</td>
</tr>
<tr>
<td></td>
<td>• <strong>SID</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>Server name</strong>: The DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Server port</strong>: The port used to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Oracle SID</strong>: The Oracle System ID (SID). To find the Oracle SID, submit the following query to your Oracle database:</td>
</tr>
</tbody>
</table>
|                    | ```sql
SELECT sys_context('userenv','instance_name') AS SID FROM dual;
``` |
<p>|                    | • <strong>Service Name</strong> |
|                    | • <strong>Server name</strong>: The DNS name or IP address of your source database server. |
|                    | • <strong>Server port</strong>: The port used to connect to your source database server. |
|                    | • <strong>Service Name</strong>: The name of the Oracle service to connect to. |
|                    | • <strong>TNS Alias</strong> |
|                    | • <strong>TNS file path</strong>: The path to the file that contains the Transparent Network Substrate (TNS) name connection information. |
|                    | • <strong>TNS file path</strong>: The TNS alias from this file to use to connect to the source database. |
|                    | • <strong>TNS Connect Identifier</strong> |</p>
<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>• TNS identifier</td>
<td>The identifier for the registered TNS connection information.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Type the user name and password to connect to your source database. <strong>Note</strong> AWS SCT uses the password to connect to your source database only when you create your project or choose the <em>Connect to source</em> option in a project, where <em>source</em> is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:</td>
</tr>
<tr>
<td>• SSL Authentication</td>
<td>Select this option to use SSL authentication for the connection.</td>
</tr>
<tr>
<td>• Trust Store</td>
<td>The location of a trust store containing certificates.</td>
</tr>
<tr>
<td>• Trust Store Password</td>
<td>The password for the trust store.</td>
</tr>
<tr>
<td>• Key Store</td>
<td>The location of a key store containing a private key and certificates. This value is required if SSL Authentication is selected and is otherwise optional.</td>
</tr>
<tr>
<td>• Trust Store Password</td>
<td>The password for the key store. This value is required if SSL Authentication is selected and is otherwise optional.</td>
</tr>
<tr>
<td><strong>Store Password</strong></td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td><strong>Oracle Driver Path</strong></td>
<td>Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8). If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10).</td>
</tr>
</tbody>
</table>

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.

---

**Using Teradata as a Source for AWS Schema Conversion Tool (AWS SCT)**

You can use AWS SCT to convert data from Teradata to Amazon Redshift.
Privileges for Teradata as a Source

The privileges required for Teradata as a source are listed following:

- SELECT ON DBC

Connecting to Teradata as a Source

Use the following procedure to connect to your Teradata source database with the AWS Schema Conversion Tool (AWS SCT).

To connect to a Teradata source database

1. In the AWS Schema Conversion Tool, choose Connect to Teradata.

   ![Connect to Teradata dialog box]

   The Connect to Teradata dialog box appears.

2. Provide the Teradata source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Type the name of the Teradata database.</td>
</tr>
<tr>
<td>For This Parameter</td>
<td>Do This</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you create your project or choose the Connect to source option in a project, where source is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td>Store Password</td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td>Encrypt Data</td>
<td>Select this option if you want to encrypt data that you exchange with the database.</td>
</tr>
<tr>
<td>Teradata Driver Path</td>
<td>Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8).</td>
</tr>
<tr>
<td></td>
<td>If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10).</td>
</tr>
</tbody>
</table>

3. Choose Test Connection to verify that you can successfully connect to your source database.
4. Choose OK to connect to your source database.

### Using LDAP Authentication with a Teradata Source

To set up Lightweight Directory Access Protocol (LDAP) authentication for Teradata users who run Microsoft Active Directory in Windows, use the following procedure.

In the procedure examples, the Active Directory domain is `test.local.com`. The Windows server is `DC`, and it is configured with default settings. The user account created in Active Directory is `test_ldap`, and the account uses the password `test_ldap`.

1. In the `/opt/teradata/tdat/tdgss/site` directory, edit the file `TdgssUserConfigFile.xml`. Change the LDAP section to the following.

```xml
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"
```

Apply the changes by running the configuration as follows.
AWS Schema Conversion Tool User Guide
Using Netezza as a Source for AWS
Schema Conversion Tool (AWS SCT)

2. Test the configuration by running the following command.

```bash
# /opt/teradata/tdat/tdgss/14.10.03.01/bin/tdsbind -u test ldap -w test ldap
```

The output should be similar to the following.

```plaintext
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
```

3. Restart TPA using the following command.

```bash
#tpareset -f "use updated TDGSSCONFIG GDO"
```

4. Create the same user in the Teradata database as in Active Directory, as shown following.

```sql
CREATE USER test_ldap AS PERM=1000, PASSWORD=test_ldap;
GRANT LOGON ON ALL TO test WITH NULL PASSWORD;
```

If you change the user password in Active Directory for your LDAP user, you should specify this new password during connection to Teradata in LDAP mode. In DEFAULT mode, you still have to connect Teradata with the LDAP user name and any password.

Using Netezza as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert data from Netezza to Amazon Redshift.
Privileges for Netezza as a Source

The privileges required for Netezza as a source are listed following:

- 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 <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

Connecting to Netezza as a Source

Use the following procedure to connect to your Netezza source database with the AWS Schema Conversion Tool (AWS SCT).

To connect to a Netezza source database

1. In the AWS Schema Conversion Tool, choose Connect to Netezza.

   ![Connect to Netezza dialog box]

   The Connect to Netezza dialog box appears.

2. Provide the Netezza source database connection information. Use the instructions in the following table.
Using Greenplum as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert data from Greenplum to Amazon Redshift.

**Privileges for Greenplum as a Source**

The privileges required for Greenplum as a source are listed following:

- `CONNECT ON DATABASE <database_name>`
- `USAGE ON SCHEMA <schema_name>`

**Connecting to Greenplum as a Source**

Use the following procedure to connect to your Greenplum source database with the AWS Schema Conversion Tool (AWS SCT).
To connect to a Greenplum source database

1. In the AWS Schema Conversion Tool, choose **Connect to Greenplum**.

   ![Connect to Greenplum dialog box](image)

   The **Connect to Greenplum** dialog box appears.

2. Provide the Greenplum source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server name</strong></td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Type the name of the Greenplum database.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
</tbody>
</table>

   **Note**

   AWS SCT uses the password to connect to your source database only when you create your project or choose the **Connect to source** option in a project, where *source* is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.
### Using Vertica as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert data from Vertica to Amazon Redshift.

#### Privileges for Vertica as a Source

The privileges required for Vertica as a source are listed following:

- USAGE ON SCHEMA `<schema_name>`
- USAGE ON SCHEMA PUBLIC
- GRANT 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)>`

#### Connecting to Vertica as a Source

Use the following procedure to connect to your Vertica source database with the AWS Schema Conversion Tool (AWS SCT).

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Verify Server Certificate</strong>: Select this option to verify the server certificate by using a trust store.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Trust Store</strong>: The location of a trust store containing certificates.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Trust Store Password</strong>: The password for the trust store.</td>
</tr>
<tr>
<td><strong>Store Password</strong></td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td><strong>Greenplum Driver Path</strong></td>
<td>Type the path to the driver to use to connect to the source database. For more information, see <strong>Installing the Required Database Drivers (p. 8)</strong>.</td>
</tr>
<tr>
<td></td>
<td>If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see <strong>Storing Driver Paths in the Global Settings (p. 10)</strong>.</td>
</tr>
</tbody>
</table>

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.
To connect to a Vertica source database

1. In the AWS Schema Conversion Tool, choose **Connect to Vertica**.

The **Connect to Vertica** dialog box appears.

2. Provide the Vertica source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Type the name of the Vertica database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
</tbody>
</table>

**Note**

AWS SCT uses the password to connect to your source database only when you create your project or choose the **Connect to source** option in a project, where **source** is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.
Using Microsoft SQL Server DW as a Source for AWS Schema Conversion Tool (AWS SCT)

You can use AWS SCT to convert data from Microsoft SQL Server DW to Amazon Redshift.

Privileges for Microsoft SQL Server Data Warehouse as a Source

The privileges required for Microsoft SQL Server data warehouse as a source are listed following:

- VIEW DEFINITION
- VIEW DATABASE STATE
- SELECT ON SCHEMA :: <schema_name>

Repeat the grant for each database whose schema you are converting.

In addition, grant the following, and run the grant on the master database:

- VIEW SERVER STATE

Connecting to SQLServerDW as a Source

Use the following procedure to connect to your Microsoft SQL Server data warehouse source database with the AWS Schema Conversion Tool (AWS SCT).
To connect to a Microsoft SQL Server data warehouse source database

1. In the AWS Schema Conversion Tool, choose Connect to Microsoft SQL Server DW.

The Connect to Microsoft SQL Server DW dialog box appears.

2. Provide the Microsoft SQL Server data warehouse source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the Domain Name Service (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Instance name</td>
<td>Type the instance name for the SQL Server database. To find the instance name, run the query SELECT @@servername; on your SQL Server database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
</tbody>
</table>

**Note**
AWS SCT uses the password to connect to your source database only when you create your project or choose the Connect to source option in a project, where source is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.
### Connecting to SQLServerDW as a Source

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this option if you want to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the <strong>SSL</strong> tab:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Trust Server Certificate</strong>: Select this option to trust the server certificate.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Trust Store</strong>: A trust store that you set up in the Global Settings.</td>
</tr>
<tr>
<td><strong>Store Password</strong></td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td><strong>Sql Server Driver Path</strong></td>
<td>Type the path to the driver to use to connect to the source database. For more information, see [Installing the Required Database Drivers](p. 8). If you store the driver path in the global project settings, the driver path doesn’t appear on the connection dialog box. For more information, see [Storing Driver Paths in the Global Settings](p. 10).</td>
</tr>
</tbody>
</table>

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.
Converting Database Schemas Using the AWS Schema Conversion Tool

You can use the AWS Schema Conversion Tool (AWS SCT) to convert your existing database schemas from one database engine to another. Converting a database using the AWS SCT user interface can be fairly simple, but there are several things you need to consider before you do the conversion.

For example, you can use the AWS SCT to do the following:

- You can also use AWS SCT to copy an existing on-premises database schema to an Amazon RDS DB instance running the same engine. You can use this feature to analyze potential cost savings of moving to the cloud and of changing your license type.
- In some cases, database features can’t be converted to equivalent Amazon RDS features. If you host and self-manage a database on the Amazon Elastic Compute Cloud (Amazon EC2) platform, you can emulate these features by substituting AWS services for them.
- AWS SCT automates much of the process of converting your online transaction processing (OLTP) database schema to an Amazon Relational Database Service (Amazon RDS) MySQL DB instance, an Amazon Aurora DB cluster, or a PostgreSQL DB instance. The source and target database engines contain many different features and capabilities, and AWS SCT attempts to create an equivalent schema in your Amazon RDS DB instance wherever possible. If no direct conversion is possible, AWS SCT provides a list of possible actions for you to take.

This section includes the following topics:

**Topics**
- Creating Mapping Rules in the AWS Schema Conversion Tool (AWS SCT) (p. 71)
- Converting Your Schema by Using the AWS Schema Conversion Tool (p. 73)
- Creating and Using the Assessment Report in the AWS Schema Conversion Tool (p. 77)
- Handling Manual Conversions in the AWS Schema Conversion Tool (p. 81)
- Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool (p. 82)
- Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 83)
- Comparing Database Schemas (p. 86)
- Finding Related Transformed Objects (p. 87)

AWS SCT supports the following OLTP conversions.

<table>
<thead>
<tr>
<th>Source Database</th>
<th>Target Database on Amazon RDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server (version 2008 and later)</td>
<td>Amazon Aurora (MySQL or PostgreSQL), Microsoft SQL Server, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>MySQL (version 5.5 and later)</td>
<td>Amazon Aurora (PostgreSQL), MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>You can migrate schema and data from MySQL to an Amazon Aurora (MySQL) DB cluster without using AWS SCT. For more information, see Migrating Data to an Amazon Aurora DB Cluster.</td>
</tr>
<tr>
<td>Oracle (version 10.2 and later)</td>
<td>Amazon Aurora (MySQL or PostgreSQL), MySQL, Oracle, PostgreSQL</td>
</tr>
</tbody>
</table>
Creating Mapping Rules in the AWS Schema Conversion Tool (AWS SCT)

Before you convert your schema with AWS SCT, you can set up rules that change the data type of columns, move objects from one schema to another, and change the names of objects. For example, if you have a set of tables in your source schema named `test_TABLE_NAME`, you can set up a rule that changes the prefix `test_` to the prefix `demo_` in the target schema.

**Note**
You can only create mapping rules if your source database engine and target database engine are different.

You can create mapping rules that perform the following tasks:

- Change data type
- Move objects
- Rename objects
- Prefix - add prefix, remove prefix, replace prefix
- Suffix - add suffix, remove suffix, replace suffix
You can create mapping rules for the following objects:

- Database
- Schema
- Table
- Column

**Creating Mapping Rules**

You can create mapping rules and save the rules as part of your project. With your project open, use the following procedure to create mapping rules.

**To create mapping rules**

1. Choose **Mapping Rules** from the **Settings** menu. The **Mapping Rules** dialog box appears.

2. Choose **Add new rule**. A new row is added to the list of rules.

3. Choose the edit icon to configure your rule.
   a. For **Name**, type a name for your rule.
   b. For **For**, choose the type of object that the rule applies to.
   c. For **where**, type a filter to apply to objects before applying the mapping rule. The where clause is evaluated by using a like clause. You can enter an exact name to select one object, or you can enter a pattern to select multiple objects.

      The fields available for the **where** clause are different depending on the type of the object. For example, if the object type is schema there is only one field available, for the schema name.

   d. For **Actions**, choose the type of mapping rule you want to create.
   e. Depending on the rule type, type one or two additional values. For example, to rename an object, type the new name of the object. To replace a prefix, type the old prefix and the new prefix.

4. After you have configured your mapping rule, choose **Save** to save your rule. You can also choose **Cancel** to cancel your changes.

5. After you are done adding, editing, and deleting rules, choose **Save All** to save all your changes.
6. Choose Close to close the Mapping Rules dialog box.

You can use the toggle icon to turn off a mapping rule without deleting it. You can use the copy icon to duplicate an existing mapping rule. You can use the delete icon to delete an existing mapping rule. To save any changes you make to your mapping rules, choose Save All.

Viewing Mapping Rules for Objects

After you set up your mapping rules, you can view the effect of the rules on specific objects in your schema before you convert your schema. In the source schema tree, choose the object you are interested in. In the main view, choose the Mapping tab. The Mapping tab opens and displays a list of all mapping rules that are applied to the object. You can see the name of the object in the source schema and the new name of the object in the target schema. If you have data type rules, you also see the data type of the column in the source schema and the new data type of the column in the target schema.

Exporting Mapping Rules

If you use AWS Database Migration Service (AWS DMS) to migrate your data from your source database to your target database, you can provide information about your mapping rules to AWS DMS. For more information about tasks, see Working with AWS Database Migration Service Replication Tasks.

To export mapping rules

1. In the AWS Schema Conversion Tool, in the source schema tree, open the context (right-click) menu and choose Export script for DMS. The save dialog box opens.
2. Browse to the location where you want to save your script, and then choose Save. Your mapping rules are saved as a JSON script that can be consumed by AWS DMS.

Converting Your Schema by Using the AWS Schema Conversion Tool

After you have connected your project to both your source database and your target Amazon RDS DB instance, your AWS Schema Conversion Tool (AWS SCT) project displays the schema from your source database in the left panel. The schema is presented in a tree-view format, and each node of the tree is lazy loaded. When you choose a node in the tree view, AWS SCT requests the schema information from your source database at that time.

You can choose schema items from your source database and then convert the schema to equivalent schema for the DB engine of your target DB instance. You can choose any schema item from your source database to convert. If the schema item that you choose depends on a parent item, then AWS SCT also generates the schema for the parent item. For example, if you choose a column from a table to convert, then AWS SCT generates the schema for the column, the table that the column is in, and the database that the table is in.

Converting Schema

To convert schema from your source database, choose a schema object to convert from the left panel of your project. Open the context (right-click) menu for the object, and then choose Convert schema, as shown following.
After you have converted the schema from your source database, you can choose schema items from the left panel of your project and view the converted schema in the center panels of your project. The lower-center panel displays the properties of and the SQL command to create the converted schema, as shown following.
After you have converted your schema, you can save your project. The schema information from your source database is saved with your project. This functionality means that you can work offline without being connected to your source database. AWS SCT connects to your source database to update the schema in your project if you choose Refresh from Database for your source database. For more information, see Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool (p. 82).

You can create a database migration assessment report of the items that can't be converted automatically. The assessment report is useful for identifying and resolving schema items that can't be converted automatically. For more information, see Creating and Using the Assessment Report in the AWS Schema Conversion Tool (p. 77).

When AWS SCT generates a converted schema, it doesn't immediately apply it to the target DB instance. Instead, the converted schema is stored locally until you are ready to apply it to the target DB instance. For more information, see Applying Your Converted Schema (p. 84).

## Editing Converted Schema

You can edit converted schema and save the changes as part of your project.

**To edit converted schema**

1. In the left panel that displays the schema from your source database, choose the schema item that you want to edit the converted schema for.
2. In the lower-center panel that displays the converted schema for the selected item, choose the SQL tab.

3. In the text displayed for the SQL tab, change the schema as needed. The schema is automatically saved with your project as you update it.

The changes that you make to converted schema are stored with your project as you make updates. If you newly convert a schema item from your source database, and you have made updates to previously converted schema for that item, those existing updates are replaced by the newly converted schema item based on your source database.

**Clearing a Converted Schema**

Until you apply the schema to your target DB instance, AWS SCT only stores the converted schema locally in your project. You can clear the planned schema from your project by choosing the tree-view node for your target DB instance, and then choosing Refresh from Database. Because no schema has been written to your target DB instance, refreshing from the database removes the planned schema elements in your AWS SCT project to match what exists in your target DB instance.
Creating and Using the Assessment Report in the AWS Schema Conversion Tool

The AWS Schema Conversion Tool (AWS SCT) creates a *database migration assessment report* to help you convert your schema. The database migration assessment report provides important information about the conversion of the schema from your source database to your target Amazon RDS DB instance. The report summarizes all of the schema conversion tasks and details the action items for schema that can’t be converted to the DB engine of your target DB instance. The report includes estimates of the amount of effort that it will take to write the equivalent code in your target DB instance that can’t be converted automatically. This Estimated Complexity field is exported in the PDF version of the assessment report, but it is not included in the CSV version.

If you are using AWS SCT to copy your existing on-premises database schema to an Amazon RDS DB instance running the same engine, the report can help you analyze requirements for moving to the AWS Cloud and for changing your license type.
Creating a Database Migration Assessment Report

Use the following procedure to create a database migration assessment report.

To create a database migration assessment report

1. In the left panel that displays the schema from your source database, choose a schema object to create an assessment report for.
2. Open the context (right-click) menu for the object, and then choose Create Report.

Assessment Report Summary

After you create an assessment report, the assessment report view opens, showing the Summary tab. The Summary tab displays the summary information from the database migration assessment report. It shows items that were converted automatically, and items that were not converted automatically.
For schema items that can’t be converted automatically to the target database engine, the summary includes an estimate of the effort required to create schema items in your target DB instance that are equivalent to those in your source.

The report categorizes the estimated time to convert these schema items as follows:

- **Simple** – Actions that can be completed in less than one hour.
- **Medium** – Actions that are more complex and can be completed in one to four hours.
- **Significant** – Actions that are very complex and take more than four hours to complete.

The section **License Evaluation and Cloud Support** contains information about moving your existing on-premises database schema to an Amazon RDS DB instance running the same engine. For example, if you want to change license types, this section of the report tells you which features from your current database should be removed.

**Assessment Report Action Items**

The assessment report view also includes an **Action Items** tab. This tab contains a list of items that can’t be converted automatically to the database engine of your target Amazon RDS DB instance. If you select an action item from the list, AWS SCT highlights the item from your schema that the action item applies to.

The report also contains recommendations for how to manually convert the schema item. For more information about deciding how to handle manual conversions, see **Handling Manual Conversions in the AWS Schema Conversion Tool (p. 81)**.
Saving the Assessment Report

You can save a local copy of the database migration assessment report as either a PDF file or a comma-separated values (CSV) file. The CSV file contains only action item information. The PDF file contains both the summary and action item information, as shown in the following example.
Handling Manual Conversions

**Database Objects with Conversion Actions for MySQL**

Of the total 179 database storage object(s) in the source database, we were able to identify 169 (94%) database storage object(s) that can be converted automatically or with minimal changes to MySQL.

10 (6%) database storage object(s) required 58 medium and 10 significant user action(s) to complete the conversion.

**Figure: Conversion statistics for database storage objects**

<table>
<thead>
<tr>
<th>Schema(1)</th>
<th>Table(19)</th>
<th>Column(177)</th>
<th>Constraint(22)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19</td>
<td>99</td>
<td>22</td>
</tr>
</tbody>
</table>

- Objects Automatically Converted
- Objects with simple actions
- Objects with medium-complexity actions
- Objects with significant actions

**Detailed Recommendations for MySQL Migrations**

If you choose to migrate your SQL Server database to MySQL, we recommend the following actions.

**Storage Object Actions**

**Constraint Changes**

Some changes are required to CONSTRAINTs that cannot be converted automatically. You'll need to address these issues manually.

Handling Manual Conversions in the AWS Schema Conversion Tool

The assessment report includes a list of items that can't be converted automatically to the database engine of your target Amazon RDS DB instance. For each item that can't be converted, there is an action item on the **Action Items** tab.

You can respond to the action items in the assessment report in the following ways:

- Modify your source database schema.
- Modify your target database schema.
Modifying Your Source Schema

For some items, it might be easier to modify the database schema in your source database to schema that can be converted automatically. First, verify that the new changes are compatible with your application architecture, then update the schema in your source database. Finally, refresh your project with the updated schema information. You can then convert your updated schema, and generate a new database migration assessment report. The action items no longer appear for the items that changed in the source schema.

The advantage of this process is that your updated schema is always available when you refresh from your source database.

Modifying Your Target Schema

For some items, it might be easier to apply the converted schema to your target database, and then add equivalent schema items manually to your target database for the items that couldn't be converted automatically. You can write all of the schema that can be converted automatically to your target DB instance by applying the schema. For more information, see Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 83).

The schema that are written to your target DB instance don't contain the items that can't be converted automatically. After applying the schema to your target DB instance, you can then manually create schema in your target DB instance that are equivalent to those in the source database. The action items in the database migration assessment report contain suggestions for how to create the equivalent schema.

**Warning**

If you manually create schema in your target DB instance, save a copy of any manual work that you do. If you apply the converted schema from your project to your target DB instance again, it overwrites the manual work you have done.

In some cases, you can't create equivalent schema in your target DB instance. You might need to rearchitect a portion of your application and database to use the functionality that is available from the DB engine for your target DB instance. In other cases, you can simply ignore the schema that can't be converted automatically.

Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool

You can update both the source schema and the target schema in your AWS Schema Conversion Tool (AWS SCT) project.

- **Source** – If you update the schema for your source database, AWS SCT replaces the schema in your project with the latest schema from your source database. Using this functionality, you can update your project if changes have been made to the schema of your source database.

- **Target** – If you update the schema for your target Amazon RDS DB instance, AWS SCT replaces the schema in your project with the latest schema from your target DB instance. If you haven't applied any schema to your target DB instance, AWS SCT clears the converted schema from your project. You can then convert the schema from your source database for a clean target DB instance.

You update the schema in your AWS SCT project by choosing **Refresh from Database**, as shown following.
Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool

When the AWS Schema Conversion Tool (AWS SCT) generates converted schema (as shown in Converting Your Schema by Using the AWS Schema Conversion Tool (p. 73)), it doesn't immediately apply the converted schema to the target DB instance. Instead, converted schema are stored locally in your project until you are ready to apply them to the target DB instance. Using this functionality, you can work with schema items that can't be converted automatically to your target DB engine. For more information on items that can't be converted automatically, see Creating and Using the Assessment Report in the AWS Schema Conversion Tool (p. 77).

You can optionally have the tool save your converted schema to a file as a SQL script prior to applying the schema to your target DB instance. You can also have the tool apply the converted schema directly to your target DB instance.

Saving Your Converted Schema to a File

You can save your converted schema as SQL scripts in a text file. By using this approach, you can modify the generated SQL scripts from AWS SCT to address items that the tool can't convert automatically. You can then run your updated scripts on your target DB instance to apply your converted schema to your target database.

To save your converted schema as SQL scripts, open the context (right-click) menu for the schema element, and choose Save as SQL, as shown following.
Applying Your Converted Schema

When you are ready to apply your converted schema to your target Amazon RDS DB instance, choose the schema element from the right panel of your project. Open the context (right-click) menu for the schema element, and then choose **Apply to database**, as shown following.
The Extension Pack Schema

The first time that you apply your converted schema to your target DB instance, AWS SCT adds an additional schema to your target DB instance. This schema implements system functions of the source database that are required when writing your converted schema to your target DB instance. The schema is called the extension pack schema.

Don't modify the extension pack schema, or you might encounter unexpected results in the converted schema that is written to your target DB instance. When your schema is fully migrated to your target DB instance, and you no longer need AWS SCT, you can delete the extension pack schema.

The extension pack schema is named according to your source database as follows:

- Microsoft SQL Server: AWS.SQLSERVER_EXT
- MySQL: AWS.MYSQL_EXT
- Oracle: AWS.ORACLE_EXT
- PostgreSQL: AWS.POSTGRESQL_EXT

For more information, see Using the AWS Lambda Functions from the AWS SCT Extension Pack (p. 142).
Comparing Database Schemas

If you made changes to your source or target schema after you migrated, you can compare the two database schemas using AWS SCT. You can compare schemas for versions the same as or earlier than the source schema.

The following schema comparisons are supported:

- Oracle to Oracle, versions 12.1.0.2.0, 11.1.0.7.0, 11.2.0.1.0, 10
- PostgreSQL to PostgreSQL and Aurora with PostgreSQL compatibility, versions 9.6, 9.5.9, 9.5.4
- MySQL to MySQL, versions 5.6.36, 5.7.17, 5.5

You specify settings for the schema comparison on the Compare Schema tab of the Project Settings page.

To compare schemas, you select the schemas, and AWS SCT indicates the objects that differ between the two schemas and the objects that don't.

**To compare two schemas**

1. Open an existing AWS SCT project, or create a project and connect to the source and target endpoints.
2. Choose the schema you want to compare.
3. Open the context (right-click) menu and choose Compare Schema.

AWS SCT indicates objects that are different between the two schemas by adding a black circle to the object's icon.
You can apply the results of the schema comparison to a single object, to a single category of objects, or to the entire schema. Choose the box next to the category, object, or schema that you want to apply the results to.

Finding Related Transformed Objects

After a schema conversion, in some cases AWS SCT might have created several objects for one schema object on the source database. For example, when performing an Oracle to PostgreSQL conversion, AWS SCT takes each Oracle trigger and transforms it into a trigger and trigger function on PostgreSQL target. Also, when AWS SCT converts an Oracle package function or procedure to PostgreSQL, it creates an equivalent function and an INIT function that should be run as init block before the procedure or function can be run.

The following procedure lets you see all related objects that were created after a schema conversion.

To view related objects that were created during a schema conversion

1. After the schema conversion, choose the converted object in the target tree view.
2. Choose the Related Converted Objects tab.
3. View the list of related target objects.
Converting Data Warehouse Schemas to Amazon Redshift by Using the AWS Schema Conversion Tool

The AWS Schema Conversion Tool (AWS SCT) automates much of the process of converting your data warehouse schema to an Amazon Redshift database schema. Because the source and target database engines can have many different features and capabilities, AWS SCT attempts to create an equivalent schema in your target database wherever possible. If no direct conversion is possible, AWS SCT provides an assessment report with a list of possible actions for you to take. Using AWS SCT, you can manage keys, map data types and objects, and create manual conversions.

AWS SCT can convert the following data warehouse schemas to Amazon Redshift.

- Greenplum Database (version 4.3 and later)
- Microsoft SQL Server (version 2008 and later)
- Netezza (version 7.0.3 and later)
- Oracle (version 10 and later)
- Teradata (version 13 and later)
- Vertica (version 7.2.2 and later)

If you want to convert an online transaction processing (OLTP) database schema, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 70).

To convert a data warehouse schema, you take the following steps.

1. Specify the optimization strategy and rules, and specify the mapping that you want AWS SCT to use. You can set up rules that change the data type of columns, move objects from one schema to another, and change the names of objects.

   You can specify optimization and mapping in Settings. For more information on optimization strategies, see Choosing Optimization Strategies and Rules for Use with the AWS Schema Conversion Tool (p. 89). For more information about mapping, see Creating Mapping Rules in the AWS Schema Conversion Tool (p. 91).

2. Provide statistics on your source data warehouse so that AWS SCT can optimize how your data warehouse is converted. You can either collect statistics directly from the database, or upload an existing statistics file. For more information about providing data warehouse statistics, see Collecting or Uploading Statistics for the AWS Schema Conversion Tool (p. 90).

3. Create a database migration assessment report that details the schema elements that can't be converted automatically. You can use this report to identify where you need to manually create a schema in your target database that is compatible with your source database. For more information about the assessment report, see Creating and Using the Assessment Report in the AWS Schema Conversion Tool (p. 98).

4. Convert the schema. AWS SCT creates a local version of the converted schema for you to review, but it doesn't apply it to your target database until you are ready. For more information about converting, see Converting Your Schema by Using the AWS Schema Conversion Tool (p. 93).
5. After you convert your schema, you can manage and edit your keys. Key management is the heart of a data warehouse conversion. For more information about managing keys, see Managing and Customizing Keys in the AWS Schema Conversion Tool (p. 97).

6. If you have schema elements that can't be converted automatically, you have two choices: update the source schema and then convert again, or create equivalent schema elements in your target database. For more information on manually converting schema elements, see Handling Manual Conversions in the AWS Schema Conversion Tool (p. 102). For more information about updating your source schema, see Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool (p. 103).

7. When you are ready, you can apply the converted schema to your target database. For more information about saving and applying the converted schema, see Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 104).

## Choosing Optimization Strategies and Rules for Use with the AWS Schema Conversion Tool

To optimize how the AWS Schema Conversion Tool (AWS SCT) converts your data warehouse schema, you can choose the strategies and rules you want the tool to use. After converting your schema, and reviewing the suggested keys, you can adjust your rules or change your strategy to get the results you want.

### To choose your optimization strategies and rules

1. Choose Settings, and then choose Project Settings. The Current project settings dialog box appears.
2. In the left pane, choose Optimization Strategies. The optimization strategies appear in the right pane with the defaults selected.
3. For Strategy Sector, choose the optimization strategy you want to use. You can choose from the following:
   
   • **Use metadata, ignore statistical information** – In this strategy, only information from the metadata is used for optimization decisions. For example, if there is more than one index on a source table, the source database sort order is used, and the first index becomes a distribution key.

   • **Ignore metadata, use statistical information** – In this strategy, optimization decisions are derived from statistical information only. This strategy applies only to tables and columns for which statistics are provided. For more information, see Collecting or Uploading Statistics for the AWS Schema Conversion Tool (p. 90).

   • **Use metadata and use statistical information** – In this strategy, both metadata and statistics are used for optimization decisions.

4. After you choose your optimization strategy, you can choose the rules you want to use. You can choose from the following:
   
   • Choose Distribution Key and Sort Keys using metadata
   • Choose fact table and appropriate dimension for collation
   • Analyze cardinality of indexes' columns
   • Find the most used tables and columns from QueryLog table
Collecting or Uploading Statistics for the AWS Schema Conversion Tool

To optimize how the AWS Schema Conversion Tool (AWS SCT) converts your data warehouse schema, you can provide statistics from your source database that the tool can use. You can either collect statistics directly from the database, or upload an existing statistics file.

To provide and review statistics

1. Open your project and connect to your source database.
2. Choose a schema object from the left panel of your project, and open the context (right-click) menu for the object. Choose Collect Statistics or Upload Statistics as shown following.
3. Choose a schema object from the left panel of your project, and then choose the Statistics tab. You can review the statistics for the object.

For each rule, you can enter a weight for the sort key and a weight for the distribution key. AWS SCT uses the weights you choose when it converts your schema. Later, when you review the suggested keys, if you are not satisfied with the results, you can return here and change your settings. For more information, see Managing and Customizing Keys in the AWS Schema Conversion Tool (p. 97).
Later, when you review the suggested keys, if you are not satisfied with the results, you can collect additional statistics and repeat this procedure. For more information, see Managing and Customizing Keys in the AWS Schema Conversion Tool (p. 97).

Creating Mapping Rules in the AWS Schema Conversion Tool

Before you convert your schema with the AWS Schema Conversion Tool (AWS SCT), you can set up rules that change the data type of columns, move objects from one schema to another, and change the names of objects. For example, if you have a set of tables in your source schema named test_TABLE_NAME, you can set up a rule that changes the prefix test_ to the prefix demo_ in the target schema.

**Note**
You can only create mapping rules if your source database engine and target database engine are different.

You can create mapping rules that perform the following tasks:
Creating Mapping Rules

You can create mapping rules and save the rules as part of your project. With your project open, use the following procedure to create mapping rules.

To create mapping rules

1. Choose **Mapping Rules** from the **Settings** menu. The **Mapping Rules** dialog box appears. The top pane contains mapping (transformation) rules.

2. In the **Transformation Rules** pane, choose **Add new rule**.

3. Configure your transformation rule.
   a. For **Name**, type a name for your rule.
   b. For **For**, choose the type of object that the rule applies to.
   c. For **where**, type a filter to apply to objects before applying the mapping rule. The where clause is evaluated by using a like clause. You can enter an exact name to select one object, or you can enter a pattern to select multiple objects.
The fields available for the where clause are different depending on the type of the object. For example, if the object type is schema there is only one field available, for the schema name.

d. For Actions, choose the type of mapping rule you want to create.

e. Depending on the rule type, type one or two additional values. For example, to rename an object, type the new name of the object. To replace a prefix, type the old prefix and the new prefix.

4. After you have configured your mapping rule, choose Save to save your rule. You can also choose Cancel to cancel your changes.

5. After you are done adding, editing, and deleting rules, choose Save All to save all your changes.

6. Choose Close to close the Mapping Rules dialog box.

You can use the toggle icon to turn off a mapping rule without deleting it. You can use the copy icon to duplicate an existing mapping rule. You can use the delete icon to delete an existing mapping rule. To save any changes you make to your mapping rules, choose Save All.

**Viewing Mapping Rules for Objects**

After you set up your mapping rules, you can view the effect of the rules on specific objects in your schema before you convert your schema. In the source schema tree, choose the object you are interested in. In the main view, choose the Mapping tab. The Mapping tab opens and displays a list of all mapping rules that are applied to the object. You can see the name of the object in the source schema and the new name of the object in the target schema. If you have data type rules, you also see the data type of the column in the source schema and the new data type of the column in the target schema.

**Exporting Mapping Rules**

If you use AWS Database Migration Service (AWS DMS) to migrate your data from your source database to your target database, you can provide information about your mapping rules to AWS DMS. For more information about tasks, see Working with AWS Database Migration Service Replication Tasks.

**To export mapping rules**

1. In the AWS Schema Conversion Tool, in the source schema tree, open the context (right-click) menu and choose Export script for DMS. The save dialog box opens.

2. Browse to the location where you want to save your script, and then choose Save. Your mapping rules are saved as a JSON script that can be consumed by AWS DMS.

**Converting Your Schema by Using the AWS Schema Conversion Tool**

After you have connected your project to both your source database and your target database, your AWS Schema Conversion Tool (AWS SCT) project displays the schema from your source database in the left panel. The schema is presented in a tree-view format, and each node of the tree is lazy loaded. When you choose a node in the tree view, AWS SCT requests the schema information from your source database at that time.

You can choose schema items from your source database and then convert the schema to equivalent schema for the database engine of your target database. You can choose any schema item from your source database to convert. If the schema item that you choose depends on a parent item, then AWS SCT also generates the schema for the parent item. For example, if you choose a column from a table
to convert, then AWS SCT generates the schema for the column, the table that the column is in, and the
database that the table is in.

Converting Schema

To convert schema from your source database, choose a schema object to convert from the left panel of
your project. Open the context (right-click) menu for the object, and then choose Convert schema, as
shown following.

After you have converted the schema from your source database, you can choose schema items from the
left panel of your project and view the converted schema in the center panels of your project. The lower-
center panel displays the properties of and the SQL command to create the converted schema, as shown
following.
After you have converted your schema, you can save your project. The schema information from your source database is saved with your project. This functionality means that you can work offline without being connected to your source database. AWS SCT connects to your source database to update the schema in your project if you choose Refresh from Database for your source database. For more information, see Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool (p. 103).

You can create a database migration assessment report of the items that can't be converted automatically. The assessment report is useful for identifying and resolving schema items that can't be converted automatically. For more information, see Creating and Using the Assessment Report in the AWS Schema Conversion Tool (p. 98).

When AWS SCT generates a converted schema, it doesn't immediately apply it to the target database. Instead, the converted schema is stored locally until you are ready to apply it to the target database. For more information, see Applying Your Converted Schema (p. 105).

**Editing Converted Schema**

You can edit converted schema and save the changes as part of your project.

**To edit converted schema**

1. In the left panel that displays the schema from your source database, choose the schema item that you want to edit the converted schema for.
2. In the lower-center panel that displays the converted schema for the selected item, choose the **SQL** tab.

3. In the text displayed for the **SQL** tab, change the schema as needed. The schema is automatically saved with your project as you update it.

The changes that you make to converted schema are stored with your project as you make updates. If you newly convert a schema item from your source database, and you have made updates to previously converted schema for that item, those existing updates are replaced by the newly converted schema item based on your source database.

### Clearing a Converted Schema

Until you apply the schema to your target database, AWS SCT only stores the converted schema locally in your project. You can clear the planned schema from your project by choosing the tree-view node for your target database, and then choosing **Refresh from Database**. Because no schema has been written to your target database, refreshing from the database removes the planned schema elements in your AWS SCT project to match what exists in your target database.
Managing and Customizing Keys in the AWS Schema Conversion Tool

After you convert your schema with the AWS Schema Conversion Tool (AWS SCT), you can manage and edit your keys. Key management is the heart of a data warehouse conversion.

To manage keys, select a table in your target database, and then choose the Key Management tab as shown following.
Creating and Using the Assessment Report in the AWS Schema Conversion Tool

The AWS Schema Conversion Tool (AWS SCT) creates a database migration assessment report to help you convert your schema. The database migration assessment report provides important information about the conversion of the schema from your source database to your target database. The report summarizes all of the schema conversion tasks and details the action items for schema that can't be converted to the DB engine of your target database. The report also includes estimates of the amount of effort that it will take to write the equivalent code in your target database that can't be converted automatically.

Creating a Database Migration Assessment Report

Use the following procedure to create a database migration assessment report.

To create a database migration assessment report

1. In the left panel that displays the schema from your source database, choose a schema object to create an assessment report for.
2. Open the context (right-click) menu for the object, and then choose Create Report.

Assessment Report Summary

After you create an assessment report, the assessment report view opens, showing the Summary tab. The Summary tab displays the summary information from the database migration assessment report. It shows items that were converted automatically, and items that were not converted automatically.
For schema items that can't be converted automatically to the target database engine, the summary includes an estimate of the effort required to create schema items in your target DB instance that are equivalent to those in your source.

The report categorizes the estimated time to convert these schema items as follows:

- **Simple** – Actions that can be completed in less than one hour.
- **Medium** – Actions that are more complex and can be completed in one to four hours.
- **Significant** – Actions that are very complex and take more than four hours to complete.

**Assessment Report Action Items**

The assessment report view also includes an *Action Items* tab. This tab contains a list of items that can't be converted automatically to the database engine of your target database. If you select an action item from the list, AWS SCT highlights the item from your schema that the action item applies to.

The report also contains recommendations for how to manually convert the schema item. For more information about deciding how to handle manual conversions, see *Handling Manual Conversions in the AWS Schema Conversion Tool* (p. 102).
Saving the Assessment Report

You can save a local copy of the database migration assessment report as either a PDF file or a comma-separated values (CSV) file. The CSV file contains only action item information. The PDF file contains both the summary and action item information, as shown in the following example.
Handling Manual Conversions in the AWS Schema Conversion Tool

The assessment report includes a list of items that can’t be converted automatically to the database engine of your target database. For each item that can’t be converted, there is an action item on the Action Items tab.

You can respond to the action items in the assessment report in the following ways:

- Modify your source database schema.
- Modify your target database schema.
Modifying Your Source Schema

For some items, it might be easier to modify the database schema in your source database to schema that can be converted automatically. First, verify that the new changes are compatible with your application architecture, then update the schema in your source database. Finally, refresh your project with the updated schema information. You can then convert your updated schema, and generate a new database migration assessment report. The action items no longer appear for the items that changed in the source schema.

The advantage of this process is that your updated schema is always available when you refresh from your source database.

Modifying Your Target Schema

For some items, it might be easier to apply the converted schema to your target database, and then add equivalent schema items manually to your target database for the items that couldn't be converted automatically. You can write all of the schema that can be converted automatically to your target database by applying the schema. For more information, see Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 104).

The schema that are written to your target database don't contain the items that can't be converted automatically. After applying the schema to your target database, you can then manually create schema in your target database that are equivalent to those in the source database. The action items in the database migration assessment report contain suggestions for how to create the equivalent schema.

Warning
If you manually create schema in your target database, save a copy of any manual work that you do. If you apply the converted schema from your project to your target database again, it overwrites the manual work you have done.

In some cases, you can't create equivalent schema in your target database. You might need to rearchitect a portion of your application and database to use the functionality that is available from the engine for your target database. In other cases, you can simply ignore the schema that can't be converted automatically.

Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool

You can update both the source schema and the target schema in your AWS Schema Conversion Tool (AWS SCT) project.

- **Source** – If you update the schema for your source database, AWS SCT replaces the schema in your project with the latest schema from your source database. Using this functionality, you can update your project if changes have been made to the schema of your source database.

- **Target** – If you update the schema for your target database, AWS SCT replaces the schema in your project with the latest schema from your target database. If you haven't applied any schema to your target database, AWS SCT clears the converted schema from your project. You can then convert the schema from your source database for a clean target database.
You update the schema in your AWS SCT project by choosing **Refresh from Database**, as shown following.

### Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool

When the AWS Schema Conversion Tool (AWS SCT) generates converted schema (as shown in Converting Your Schema by Using the AWS Schema Conversion Tool (p. 93)), it doesn't immediately apply the converted schema to the target database. Instead, converted schema are stored locally in your project until you are ready to apply them to the target database. Using this functionality, you can work with schema items that can't be converted automatically to your target database engine. For more information on items that can't be converted automatically, see Creating and Using the Assessment Report in the AWS Schema Conversion Tool (p. 98).

You can optionally have the tool save your converted schema to a file as a SQL script prior to applying the schema to your target database. You can also have the tool apply the converted schema directly to your target database.

#### Saving Your Converted Schema to a File

You can save your converted schema as SQL scripts in a text file. By using this approach, you can modify the generated SQL scripts from AWS SCT to address items that the tool can't convert automatically. You can then run your updated scripts on your target database to apply your converted schema to your target database.
To save your converted schema as SQL scripts, open the context (right-click) menu for the schema element, and choose **Save as SQL**, as shown following.

Applying Your Converted Schema

When you are ready to apply your converted schema to your target database, choose the schema element from the right panel of your project. Open the context (right-click) menu for the schema element, and then choose **Apply to database**, as shown following.
The Extension Pack Schema

The first time that you apply your converted schema to your target DB instance, AWS SCT adds an additional schema to your target DB instance. This schema implements system functions of the source database that are required when writing your converted schema to your target DB instance. The schema is called the extension pack schema.

Don't modify the extension pack schema, or you might encounter unexpected results in the converted schema that is written to your target DB instance. When your schema is fully migrated to your target DB instance, and you no longer need AWS SCT, you can delete the extension pack schema.

The extension pack schema is named according to your source database as follows:

- 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

For more information, see Using the AWS Schema Conversion Tool Extension Pack (p. 140).
Python Libraries

To create custom functions in Amazon Redshift, you use the Python language. Use the AWS SCT extension pack to install python libraries for your Amazon Redshift database. For more information, see Using the AWS Schema Conversion Tool Extension Pack (p. 140).

Optimizing Amazon Redshift by Using the AWS Schema Conversion Tool

You can use the AWS Schema Conversion Tool (AWS SCT) to optimize your Amazon Redshift database. Using your Amazon Redshift database as a source, and a test Amazon Redshift database as the target, AWS SCT recommends sort keys and distribution keys to optimize your database.

Optimizing Your Amazon Redshift Database

Use the following procedure to optimize your Amazon Redshift database.

To optimize your Amazon Redshift database

1. Take a manual snapshot of your Amazon Redshift cluster as a backup. You can delete the snapshot after you are done optimizing your Amazon Redshift cluster and testing any changes that you make. For more information, see Amazon Redshift Snapshots.

2. Choose a schema object to convert from the left panel of your project. Open the context (right-click) menu for the object, and then choose Collect Statistics.

   AWS SCT uses the statistics to make suggestions for sort keys and distribution keys.

3. Choose a schema object to optimize from the left panel of your project. Open the context (right-click) menu for the object, and then choose Run Optimization.

   AWS SCT makes suggestions for sort keys and distribution keys.

4. To review the suggestions, expand the tables node under your schema in the left panel of your project, and then choose a table. Choose the Key Management tab as shown following.
The left pane contains key suggestions, and includes the confidence rating for each suggestion. You can choose one of the suggestions, or you can customize the key by editing it in the right pane.

5. You can create a report that contains the optimization suggestions. To create the report, do the following:

   a. Choose a schema object that you optimized from the left panel of your project. Open the context (right-click) menu for the object, and then choose Create Report.

      The report opens in the main window, and the Summary tab appears. The number of objects with optimization suggestions appears in the report.

   b. Choose the Action Items tab to see the key suggestions in a report format.

   c. You can save a local copy of the optimization report as either a PDF file or a comma-separated values (CSV) file. The CSV file contains only action item information. The PDF file contains both the summary and action item information.

6. To apply suggested optimizations to your database, choose an object in the right panel of your project. Open the context (right-click) menu for the object, and then choose Apply to database.
Using the AWS Schema Conversion Tool with the AWS Database Migration Service

Using an AWS SCT Replication Agent with AWS DMS

For very large database migrations, you can use an AWS SCT replication agent to copy data from your on-premises database to Amazon S3 or an Amazon Snowball device. The replication agent works in conjunction with AWS DMS, and the replication agent can work in the background while AWS SCT is closed.

When working with Amazon Snowball, the AWS SCT agent extracts data to the Amazon Snowball device. The device is then sent to AWS and the data is loaded to an Amazon S3 bucket. During this time, the AWS SCT agent continues to run. The agent then takes the data on Amazon S3 and copies the data to the target endpoint.

For more information, see the AWS DMS documentation.
Using Data Extraction Agents

You can use the AWS SCT data extraction agents to extract data from your on-premises data warehouse and migrate it to Amazon Redshift. The extraction agent extracts your data and uploads the data to either Amazon S3 or an AWS Snowball device. You can then use AWS SCT to copy the data to Amazon Redshift.

Amazon S3 is a storage and retrieval service. To store an object in Amazon S3, you upload the file you want to store to an Amazon S3 bucket. When you upload a file, you can set permissions on the object and also on any metadata.

Large-scale data migrations can include many terabytes of information, and can be slowed by network performance and by the sheer amount of data that has to be moved. AWS Snowball is an AWS service you can use to transfer data to the cloud at faster-than-network speeds using an AWS-owned appliance. An AWS Snowball device can hold up to 80 TB of data and an AWS Snowball Edge device can hold up to 100 TB of data. It uses 256-bit encryption and an industry-standard Trusted Platform Module (TPM) to ensure both security and full chain-of-custody for your data. AWS SCT works with both AWS Snowball devices and AWS Snowball Edge devices, referred to as AWS Snowball devices in the rest of this guide.

When you use AWS SCT and an AWS Snowball device, you migrate your data in two stages. First, you use the AWS SCT to process the data locally and then move that data to the AWS Snowball device. You then send the device to AWS using the AWS Snowball process, and then AWS automatically loads the data into an Amazon S3 bucket. Next, when the data is available on Amazon S3, you use AWS SCT to migrate the data to Amazon Redshift. Data extraction agents can work in the background while AWS SCT is closed.

The following diagram shows the supported scenario.
Data extraction agents are currently supported for the following source data warehouses:

- Greenplum Database (version 4.3 and later)
- Microsoft SQL Server (version 2008 and later)
- Netezza (version 7.0.3 and later)
- Oracle (version 10 and later)
- Teradata (version 13 and later)
- Vertica (version 7.2.2 and later)

You can connect to FIPS endpoints for Amazon Redshift if you need to comply with the Federal Information Processing Standard security requirements. FIPS endpoints are available in the following AWS Regions:

- US East (N. Virginia) Region (redshift-fips.us-east-1.amazonaws.com)
- US East (Ohio) Region (redshift-fips.us-east-2.amazonaws.com)
- US West (N. California) Region (redshift-fips.us-west-1.amazonaws.com)
- US West (Oregon) Region (redshift-fips.us-west-2.amazonaws.com)

Use the information in the following topics to learn how to work with data extraction agents.

Topics

- Prerequisite Settings for Amazon S3 and Security for Data Extraction Agents (p. 111)
- Installing Extraction Agents (p. 113)
- Registering Extraction Agents with the AWS Schema Conversion Tool (p. 115)
- Hiding and Recovering Information for an AWS SCT Agent (p. 116)
- Creating Data Extraction Filters in the AWS Schema Conversion Tool (p. 117)
- Sorting Data Before Migrating Using AWS SCT (p. 118)
- Creating, Running, and Monitoring an AWS SCT Data Extraction Task (p. 119)
- Data Extraction Using a AWS Snowball Device (p. 121)
- Data Extraction Task Output (p. 127)
- Using Virtual Partitioning with AWS Schema Conversion Tool (p. 128)
- Migrating LOBs to Amazon Redshift (p. 131)
- Best Practices and Troubleshooting for Data Extraction Agents (p. 132)

Prerequisite Settings for Amazon S3 and Security for Data Extraction Agents

Before you work with data extraction agents, store your Amazon S3 bucket information and set up your Secure Sockets Layer (SSL) trust and key store.

Amazon S3 Settings

After your agents extract your data, they upload it to your Amazon S3 bucket. Before you continue, you must provide the credentials to connect to your AWS account and your Amazon S3 bucket. You store
your credentials and bucket information in a profile in the global application settings, and then associate
the profile with your AWS SCT project. If necessary, choose **Global Settings** to create a new profile. For
more information, see **Using AWS Service Profiles in the AWS Schema Conversion Tool** (p. 13).

## Security Settings

The AWS Schema Conversion Tool and the extraction agents can communicate through Secure Sockets
Layer (SSL). To enable SSL, set up a trust store and key store.

### To set up secure communication with your extraction agent

1. Start the AWS Schema Conversion Tool.
2. Open the **Settings** menu, and then choose **Global Settings**. The **Global settings** dialog box appears.

Choose the **Security** tab as shown following.

3. Choose **Generate Trust and Key Store**, or choose **Select existing Trust and Key Store**.

   If you choose **Generate Trust and Key Store**, you then specify the name and password for the trust
   and key stores, and the path to the location for the generated files. You use these files in later steps.

   If you choose **Select existing Trust and Key Store**, you then specify the password and file name for
   the trust and key stores. You use these files in later steps.

4. After you have specified the trust store and key store, choose **OK** to close the **Global Settings** dialog box.
Installing Extraction Agents

We recommend that you install multiple extraction agents on individual computers, separate from the computer that is running the AWS Schema Conversion Tool.

Extraction agents are currently supported on the following operating systems:

- macOS
- Microsoft Windows
- Red Hat Enterprise Linux (RHEL) 6.0
- Ubuntu Linux (version 14.04 and later)

Use the following procedure to install extraction agents. Repeat this procedure for each computer that you want to install an extraction agent on.

To install an extraction agent

1. If you have not already downloaded the AWS SCT installer file, follow the instructions at Installing, Verifying, and Updating the AWS Schema Conversion Tool (AWS SCT) (p. 4) to download it. The .zip file that contains the AWS SCT installer file also contains the extraction agent installer file.
2. Locate the installer file for your extraction agent in a subfolder named agents. For each computer operating system, the correct file to install the extraction agent is shown following.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>macOS</td>
<td>aws-schema-conversion-tool-extractor-1.0.build-number.dmg</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>aws-schema-conversion-tool-extractor-1.0.build-number.msi</td>
</tr>
<tr>
<td>RHEL</td>
<td>aws-schema-conversion-tool-extractor-1.0.build-number.x86_64.rpm</td>
</tr>
<tr>
<td>Ubuntu Linux</td>
<td>aws-schema-conversion-tool-extractor-1.0.build-number.deb</td>
</tr>
</tbody>
</table>

3. To install the extraction agent on a separate computer, copy the installer file to the new computer.
4. Run the installer file. Use the instructions for your operating system, shown following.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Installation Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>macOS</td>
<td>In Finder, open aws-schema-conversion-tool-extractor-1.0.build-number.dmg. Drag aws-schema-conversion-tool-extractor-1.0.build-number.dmg to the Applications folder.</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>Double-click the file to run the installer.</td>
</tr>
</tbody>
</table>
| RHEL                  | Run the following command in the folder that you downloaded or moved the file to:  
                            sudo rpm -ivh aws-schema-conversion-tool-extractor-1.0.build-number.x86_64.rpm |
### Operating System | Installation Instructions
--- | ---
Ubuntu Linux | Run the following command in the folder that you downloaded or moved the file to:

```
sudo dpkg -i aws-schema-conversion-tool-extractor-1.0.build-number.deb
```

5. Install the Java Database Connectivity (JDBC) drivers for your source database engine. For instructions and download links, see  [Installing the Required Database Drivers (p. 8)](#). Follow the instructions for your source database engine only, not your target database engine.

6. Copy the SSL trust and key stores (.zip or individual files) that you generated in an earlier procedure. If you copy the .zip file to a new computer, extract the individual files from the .zip file on the new computer.

You can put the files anywhere you want. However, note the locations because in a later procedure you tell the agent where to find the files.

Continue installing your extraction agent by completing the procedure in the following section.

## Configuring Extraction Agents

Use the following procedure to configure extraction agents. Repeat this procedure on each computer that has an extraction agent installed.

### To configure your extraction agent

- From the location where you installed the agent, run the setup program. For RHEL and Ubuntu, the file is named `sct-extractor-setup.sh`. For macOS and Microsoft Windows, the file is named AWS SCT Data Extractor Agent, and you can double-click the file to run it.

  The setup program prompts you for information. For each prompt, a default value appears. You can accept the default value, or type a new value. You specify the following information:

  - The data warehouse engine.
  - The port number the agent listens on.
  - The location where you installed the JDBC drivers.
  - The working folder. Your extracted data goes into a subfolder of this location. The working folder can be on a different computer from the agent, and a single working folder can be shared by multiple agents on different computers.
  - The location of the key store file.
  - The password for the key store.
  - The location of the trust store file.
  - The password for the trust store.

  The setup program updates the settings file for the extraction agent. The settings file is named `Settings.properties`, and is located where you installed the extraction agent. The following is a sample settings file.

```properties
port=8888
vendor=ORACLE
driver.jars=<driver path>/Install/Drivers/ojdbc7.jar
location=<output path>/dmf/8888/out
extractor.log.folder=<log path>/dmf/8888/log
extractor.storage.folder=<storage path>/dmf/8888/storage
```
Starting Extraction Agents

Use the following procedure to start extraction agents. Repeat this procedure on each computer that has an extraction agent installed.

Extraction agents act as listeners. When you start an agent with this procedure, the agent starts listening for instructions. You send the agents instructions to extract data from your data warehouse in a later section.

**To start your extraction agent**

- On the computer that has the extraction agent installed, run the command listed following for your operating system.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Start Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>macOS</td>
<td>Run the StartAgent.command file.</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>Double-click the StartAgent.bat batch file.</td>
</tr>
</tbody>
</table>
| RHEL               | Run the following command in the path to the folder that you installed the agent:  
                    |  sudo initctl start sct-extractor                                             |
| Ubuntu Linux       | Run the following command in the path to the folder that you installed the agent. Use the command appropriate for your version of Ubuntu.  
                    |  Ubuntu 14.04: sudo initctl start sct-extractor                               |
|                    |  Ubuntu 15.04 and later: sudo systemctl start sct-extractor                   |

To check the status of the agent, run the same command but replace `start` with `status`.

To stop an agent, run the same command but replace `start` with `stop`.

Registering Extraction Agents with the AWS Schema Conversion Tool

You manage your extraction agents by using AWS SCT. The extraction agents act as listeners. When they receive instructions from AWS SCT, they extract data from your data warehouse.

Use the following procedure to register extraction agents with your AWS SCT project.

**To register an extraction agent**

1. Start the AWS Schema Conversion Tool, and open a project.
2. Open the View menu, and then choose Data Migration View. The Agents tab appears. If you have previously registered agents, they appear in a grid at the top of the tab as shown following.

<table>
<thead>
<tr>
<th>Description</th>
<th>Version</th>
<th>Host Name</th>
<th>Port</th>
<th>Status</th>
<th>Uptime</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent3ubuntu</td>
<td>1.01</td>
<td>ec2-54-19...</td>
<td>8192</td>
<td>✔️</td>
<td>0h:11m</td>
</tr>
<tr>
<td>agent2rhel</td>
<td>1.01</td>
<td>ec2-54-24...</td>
<td>8192</td>
<td>✔️</td>
<td>0h:12m</td>
</tr>
<tr>
<td>agent1mac</td>
<td>1.01</td>
<td>c4b301bf6...</td>
<td>8192</td>
<td>✔️</td>
<td>0h:1m</td>
</tr>
</tbody>
</table>

3. Choose Register. The New Agent Registration dialog box appears.

   **Note**
   After you register an agent with an AWS SCT project, you can't register the same agent with a different project. If you're no longer using an agent in an AWS SCT project, you can unregister it. You can then register it with a different project.

4. Enter your information in the New Agent Registration dialog box:
   a. For **Description**, type a description of the agent.
   b. For **Host Name**, type the host name or IP address of the computer of the agent.
   c. For **Port**, type the port number that the agent is listening on.
   d. Choose **Register** to register the agent with your AWS SCT project.

5. Repeat the previous steps to register multiple agents with your AWS SCT project.

---

**Hiding and Recovering Information for an AWS SCT Agent**

An AWS SCT agent encrypts a significant amount of information, for example passwords to user key-trust stores, database accounts, AWS account information, and similar items. It does so using a special file called seed.dat. By default, the agent creates this file in the working folder of the user who first configures the agent.

Because different users can configure and run the agent, the path to seed.dat is stored in the {extractor.private.folder} parameter of the settings.properties file. When the agent
Creating Data Extraction Filters in the AWS Schema Conversion Tool

Before you extract your data with the AWS Schema Conversion Tool (AWS SCT), you can set up filters that reduce the amount of data that you extract. You can create data extraction filters by using

Version 1.0

117
clauses to reduce the data that you extract. For example, you can write a WHERE clause that selects data from a single table.

You can create data extraction filters and save the filters as part of your project. With your project open, use the following procedure to create data extraction filters.

**To create data extraction filters**

2. In the Filtering Rules pane, choose Add new rule.
3. Configure your filter:
   a. For Name, type a name for your filter.
   b. For Where schema name like, type a filter to apply to schemas. In this filter, a WHERE clause is evaluated by using a LIKE clause. You can enter an exact name to choose one schema, or you can enter a pattern to choose multiple schemas.
   c. For table name like, type a filter to apply to tables. In this filter, a WHERE clause is evaluated by using a LIKE clause. You can enter an exact name to choose one table, or you can enter a pattern to choose multiple tables.
   d. For Where clause, type a WHERE clause to filter data.
4. After you have configured your filter, choose Save to save your filter, or Cancel to cancel your changes.
5. After you are done adding, editing, and deleting filters, choose Save All to save all your changes, and then choose Close.

To turn off a filter without deleting it, use the toggle icon. To duplicate an existing filter, use the copy icon. To delete an existing filter, use the delete icon. To save any changes you make to your filters, choose Save All.

**Sorting Data Before Migrating Using AWS SCT**

Sorting your data before migration with AWS SCT provides some benefits. If you sort data first, AWS SCT can restart the extraction agent at the last saved point after a failure. Also, if you are migrating data to Amazon Redshift and you sort data first, AWS SCT can insert data into Amazon Redshift faster.

These benefits have to do with how AWS SCT creates data extraction queries. In some cases, AWS SCT uses the DENSE_RANK analytic function in these queries. However, DENSE_RANK can use lots of time and server resources to sort the dataset that results from extraction, so if AWS SCT can work without it, it does.

**To sort data before migrating using AWS SCT**

1. Open an AWS SCT project.
2. Open the context (right-click) menu for the object, and then choose Create Local Task.
3. Choose the Advanced tab, and for Sorting Strategy, choose an option:
   - **Never use sorting** – The extraction agent doesn’t use the DENSE_RANK analytic function and restarts from the beginning if a failure occurs.
   - **Use sorting if possible** – The extraction agent uses DENSE_RANK if the table has a primary key or a unique constraint.
Creating, Running, and Monitoring an AWS SCT Data Extraction Task

Use the following procedures to create, run, and monitor data extraction tasks.

To assign tasks to agents and migrate data

1. In the AWS Schema Conversion Tool, after you have converted your schema, choose one or more tables from the left panel of your project.

You can choose all tables, but we recommend against that for performance reasons. We recommend that you create multiple tasks for multiple tables based on the size of the tables in your data warehouse.
2. Open the context (right-click) menu for each table, and then choose Create Task. The Create Local Task dialog box opens, as shown following.

3. For Task Name, type a name for the task.

4. For Migration Mode, choose one of the following:
   - Extract Only – Extract your data, and save the data to your local working folders.
   - Extract and Upload – Extract your data, and upload your data to Amazon S3.
   - Extract, Upload and Copy – Extract your data, upload your data to Amazon S3, and copy it into your Amazon Redshift data warehouse.

5. Choose Extract LOBs to extract large objects. If you don't need to extract large objects, you can clear the check box. Doing this reduces the amount of data that you extract.

6. If you want to see detailed information about a task, choose Enable Task Logging. You can use the task log to debug problems.

   If you enable task logging, choose the level of detail that you want to see. The levels are the following, with each level including all messages from the previous level:
   - ERROR – The smallest amount of detail.
   - WARNING
   - INFO
   - DEBUG
   - TRACE – The largest amount of detail.

7. Choose Test Task to verify that you can connect to your working folder, Amazon S3 bucket, and Amazon Redshift data warehouse. The verification depends on the migration mode you chose.

8. Choose Create to create the task.

9. Repeat the previous steps to create tasks for all the data that you want to migrate.

To run and monitor tasks
1. For View, choose Data Migration View. The Agents tab appears.
2. Choose the Tasks tab. Your tasks appear in the grid at the top as shown following. You can see the status of a task in the top grid, and the status of its subtasks in the bottom grid.

3. Choose a task in the top grid and expand it. Depending on the migration mode you chose, you see the task divided into Extract, Upload, and Copy.

4. Choose Start for a task to start that task. You can monitor the status of your tasks while they work. The subtasks run in parallel. The extract, upload, and copy also run in parallel.

5. If you enabled logging when you set up the task, you can view the log:
   a. Choose Download Log. A message appears with the name of the folder that contains the log file. Dismiss the message.
   b. A link appears in the Task details tab. Choose the link to open the folder that contains the log file.

You can close AWS SCT, and your agents and tasks continue to run. You can reopen AWS SCT later to check the status of your tasks and view the task logs.

Data Extraction Using a AWS Snowball Device

The process of using AWS SCT and AWS Snowball has several steps. The migration involves a local task, where AWS SCT uses a data extraction agent to move the data to the AWS Snowball device, then an intermediate action where AWS copies the data from the AWS Snowball device to an Amazon S3 bucket. The process finishes AWS SCT loading the data from the Amazon S3 bucket to Amazon Redshift.

The sections following this overview provide a step-by-step guide to each of these tasks. The procedure assumes that you have AWS SCT installed and that you have configured and registered a data extraction agent on a dedicated machine.

The following steps need to occur to migrate data from a local data store to an AWS data store using AWS Snowball.
1. Create an AWS Snowball job using the AWS Snowball console. For more information, see Create an Import Job in the AWS Snowball documentation.
2. Unlock the AWS Snowball Edge device using the local, dedicated Linux machine.
3. Create a new project in AWS SCT using the registered data extraction agent.
4. Install the database driver for your source database on the dedicated machine where you installed the data extractor.
5. Create and set permissions for the Amazon S3 bucket to use.
6. Create Local & DMS Task in SCT.
7. Run and monitor the Local & DMS Task in SCT.
8. Run the AWS SCT task and monitor progress in SCT.

Step-by-Step Procedures for Migrating Data Using AWS SCT and AWS Snowball

The following sections provide detailed information on the migration steps.

Step 1: Create an AWS Snowball Job

Create an AWS Snowball job by following the steps outlined in the section Getting Started with AWS Snowball Edge: Your First Job in the AWS Snowball documentation.

Step 2: Unlock the AWS Snowball Edge Device

You should run the commands that unlock and provide credentials to the Snowball Edge device from the machine where you installed the DMS Agent. This way you can be sure that the DMS Agent call connect to the AWS Snowball Edge device. For more information about unlocking the AWS Snowball Edge device, see Unlock the Snowball Edge.

For example, the following command lists the Amazon S3 bucket used by the device.

```bash
aws s3 ls s3://<bucket-name> --profile <Snowball Edge profile> --endpoint http://<Snowball IP>:8080 --recursive
```

Step 3: Create a New AWS SCT Project

Next, you create a new AWS SCT project.

To create a new project in AWS SCT

2. Add the following project information.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>Type a name for your project, which is stored locally on your computer.</td>
</tr>
<tr>
<td>Location</td>
<td>Type the location for your local project file.</td>
</tr>
<tr>
<td>OLTP</td>
<td>Choose Transactional Database (OLTP).</td>
</tr>
</tbody>
</table>
### For This Parameter | Do This
---|---
Source DB Engine | Choose your source data store.
Target DB Engine | Choose your target data store.

3. Choose OK to create your AWS SCT project.
4. (Optional) Test your connection.

## Step 4: Install the Source Database Driver for the DMS Agent on the Linux Computer

For the migration to succeed, the DMS Agent must be able to connect to the source database. To make this possible, you install the database driver for your source database. The required driver varies by database.

To restart the DMS Agent after database driver installation, change the working directory to `<product_dir>/bin` and use the steps listed following for each source database.

```
cd <product_dir>/bin
./arep.ctl stop
./arep.ctl start
```

### To install on Oracle

Install Oracle Instant Client for Linux (x86-64) version 11.2.0.3.0 or later.

In addition, if not already included in your system, you need to create a symbolic link in the `$ORACLE_HOME\lib` directory. This link should be called libclntsh.so, and should point to a specific version of this file. For example, on an Oracle 12c client:

```
lrwxrwxrwx 1 oracle oracle 63 Oct 2 14:16 libclntsh.so ->
/u01/app/oracle/home/lib/libclntsh.so.12.1
```

In addition, the `LD_LIBRARY_PATH` environment variable should be appended with the Oracle lib directory and added to the `site_arep_login.sh` script under the lib folder of the installation. Add this script if it doesn’t exist.

```
vi cat <product dir>/bin/site_arep_login.sh
```

```
export ORACLE_HOME=/usr/lib/oracle/12.2/client64; export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

### To install on Microsoft SQL Server

Install the Microsoft ODBC Driver

To install the Simba ODBC Driver

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/microsoft/msodbcsql/lib64/
```
Edit the simba.sqlserverodbc.ini file as follows

DriverManagerEncoding=UTF-16
ODBCInstLib=libodbcinst.so

To install on SAP Sybase

The SAP Sybase ASE ODBC 64-bit client should be installed

If the installation dir is /opt/sap, update the site_arep_login.sh with

```bash
export SYBASE_HOME=/opt/sap
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SYBASE_HOME/      
  DataAccess64/ODBC/lib:$SYBASE_HOME/DataAccess/ODBC/      
  lib:$SYBASE_HOME/OC5-16.0/lib:$SYBASE_HOME/OC5-16.0/      
  lib3p64:$SYBASE_HOME/OC5-16.0/lib3p
```

The /etc/odbcinst.ini should include these entries

```ini
[Sybase]
Driver=/opt/sap/DataAccess64/ODBC/lib/libsybdrvodb.so
Description=Sybase ODBC driver
```

To install on MySQL

Install MySQL Connector/ODBC for Linux, version 5.2.6 or later

Make sure that the /etc/odbcinst.ini file contains an entry for MySQL, as in the following example

```ini
[MySQL ODBC 5.2.6 Unicode Driver]
Driver = /usr/lib64/libmyodbc5w.so
UsageCount = 1
```

To install on PostgreSQL

Install postgresql94-9.4.4-1PGDG.<OS Version>.x86_64.rpm. This is the package that contains the psql executable.

For example, postgresql94-9.4.4-1PGDG.rhel7.x86_64.rpm is the package required for Red Hat 7.

Install the ODBC driver postgresql94-odbc-09.03.0400-1PGDG.<OS version>.x86_64 or above for Linux, where <OS version> is the OS of the agent machine.

For example, postgresql94-odbc-09.03.0400-1PGDG.rhel7.x86_64 is the client required for Red Hat 7.

Make sure that the /etc/odbcinst.ini file contains an entry for PostgreSQL, as in the following example

```ini
[PostgreSQL]
Description = PostgreSQL ODBC driver
Driver = /usr/pgsql-9.4/lib/psqlodbc.so
Setup = /usr/pgsql-9.4/lib/psqlodbcw.so
```
Step 5: Configure AWS SCT to Access the Amazon S3 Bucket

For information on configuring an Amazon S3 bucket, see Working with Amazon S3 Buckets in the Amazon S3 documentation.

Step 6: Creating a Local & DMS Task

Next, you create the task that is the end-to-end migration task. The task includes two subtasks. One subtask migrates data from the source database to the AWS Snowball appliance. The other subtask takes the data that the appliance loads into an Amazon S3 bucket and migrates it to the target database.

To create the end-to-end migration task

1. Start AWS SCT, choose View, and then choose Database Migration View (Local & DMS).

2. In the left panel that displays the schema from your source database, choose a schema object to migrate. Open the context (right-click) menu for the object, and then choose Create Local & DMS Task.

3. Add your task information.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Name</td>
<td>Type a name for the task.</td>
</tr>
<tr>
<td>Agent</td>
<td>Choose DMS Agent.</td>
</tr>
</tbody>
</table>
### For This Parameter | Do This
--- | ---
**Replication Instance** | Choose the AWS DMS replication instance that you want to use.

**Migration Type** | Choose the type of migration you want.

- **Migrate existing data** to migrate the contents of the chosen schema. This process is called a full load in AWS DMS.
- **Migrate existing data and replicate ongoing changes** to migrate the contents of the chosen schema and capture all ongoing changes to the database. This process is called full load and CDC in AWS DMS.

**Target table preparation mode** | Choose the preparation mode you want to use.

- **Truncate** - Tables are truncated without affecting table metadata.
- **Drop tables on target** - The tables are dropped and new tables are created in their place.
- **Do nothing** - Data and metadata of the target tables are not changed.

**IAM role** | Choose the predefined IAM role that has permissions to access the Amazon S3 bucket and the target database. For more information about the permissions required to access an Amazon S3 bucket, see Amazon S3 Settings (p. 111).

**Logging** | Choose **Enable** to have AWS CloudWatch create logs for the migration. You incur charges for this service. For more information about AWS CloudWatch, see How Amazon CloudWatch Works.

**Description** | Type a description of the task.

**Use Snowball** | Choose this check box to use Snowball.

**Job Name** | Choose the AWS Snowball job name you created.

**Snowball IP** | Type the IP address of the AWS Snowball appliance.

**Port** | Type the port value for the AWS Snowball appliance.

**Local AWS S3 Access key** | Type the AWS access key for the account you are using for the migration.

**Local AWS S3 Secret key** | Type the AWS secret key for the account you are using for the migration.

4. Choose **Create** to create the task.

---

**Step 7: Running and Monitoring the SCT Task**

You can start the Local & DMS Task when all connections to endpoints are successful. This means all connections for the Local task, which includes connections from the DMS Agent to the source database,
the staging Amazon S3 bucket, and the AWS Snowball device, as well as the connections for the DMS task, which includes connections from the staging Amazon S3 bucket to the target database on AWS.

You can monitor the DMS Agent logs by choosing Show Log. The log details include agent server (Agent Log) and local running task (Task Log) logs. Because the endpoint connectivity is done by the server (since the local task is not running and there are no task logs), connection issues are listed under the Agent Log tab.

<table>
<thead>
<tr>
<th>Migration Mode</th>
<th>Data Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract, Upload and Copy</td>
<td>The data is already in your Amazon Redshift data warehouse. You can verify that the data is there, and start using it. For more information, see Connecting to Clusters From Client Tools and Code.</td>
</tr>
<tr>
<td>Extract and Upload</td>
<td>The extraction agents saved your data as files in your Amazon S3 bucket. You can use the Amazon Redshift COPY command to load your data to Amazon Redshift. For more information, see Loading Data from Amazon S3 in the Amazon Redshift documentation.</td>
</tr>
<tr>
<td></td>
<td>There are multiple folders in your Amazon S3 bucket, corresponding to the extraction tasks that you set up. When you load your data to Amazon Redshift, specify the name of the manifest file created by each task. The manifest file appears in the task folder in your S3 bucket as shown following.</td>
</tr>
</tbody>
</table>
Extract Only

The extraction agents saved your data as files in your working folder. Manually copy your data to your Amazon S3 bucket, and then proceed with the instructions for Extract and Upload.

### Using Virtual Partitioning with AWS Schema Conversion Tool

You can often best manage large non-partitioned tables by creating subtasks that create virtual partitions of the table data using filtering rules. In AWS SCT, you can create virtual partitions for your migrated data. There are three partition types, which work with specific data types:

- The RANGE partition type works with numeric and date and time data types.
- The LIST partition type works with numeric, character, and date and time data types.
- The DATE AUTO SPLIT partition type works with date and time data types.
AWS SCT validates the values you provide for creating a partition. For example, if you attempt to partition a column with data type NUMERIC but you provide values of a different data type, AWS SCT throws an error.

**Limits When Creating Virtual Partitioning**

These are limitations to creating a virtual partition:

- You can only use virtual partitioning only for nonpartitioned tables.
- You can use virtual partitioning only in the data migration view.
- You can't use the option UNION ALL VIEW with virtual partitioning.

**RANGE Partition Type**

The RANGE partition type partitions data based on a range of column values for numeric and date and time data types. This partition type creates a `WHERE` clause, and you provide the range of values for each partition. You specify a list of values for the partitioned column in the box **Values**. You can load value information by using a .csv file.

For example, you can create multiple partitions based on a value range you provide. In the following example, the partitioning values for LO_TAX are specified to create multiple partitions.

| 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 |

**To create a RANGE virtual partition**

1. Open the AWS SCT application.
2. Choose **Data Migration View** mode.
3. Choose the table where you want to set up virtual partitioning. Open the context (right-click) menu for the table, and choose **Add Virtual Partitioning**.
4. In the **Add Virtual Partitioning** dialog box, enter the information as follows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partition Type</strong></td>
<td>Choose <strong>RANGE</strong>. The dialog box UI changes depending on the type you choose.</td>
</tr>
<tr>
<td><strong>Column Name</strong></td>
<td>Choose the column that you want to partition.</td>
</tr>
<tr>
<td><strong>Column Type</strong></td>
<td>Choose the data type for the values in the column.</td>
</tr>
<tr>
<td><strong>Values</strong></td>
<td>Add new values by typing each value in the <strong>New Value</strong> box, then choosing the plus sign to add the value.</td>
</tr>
<tr>
<td><strong>Load From File</strong></td>
<td>(Optional) Type the name of a .csv file that contains the partition values.</td>
</tr>
</tbody>
</table>

5. Choose **OK**.

**LIST Partition Type**

The LIST partition type partitions data based on column values for numeric, character, and date and time data types. This partition type creates a `WHERE` clause, and you provide the values for each partition. You
specify a list of values for the partitioned column in the field **Values**. You can load value information by using a .csv file.

For example, you can create multiple partitions based on a value you provide. In the following example, the partitioning values for LO_ORDERKEY are specified to create multiple partitions.

| Partition1: | WHERE LO_ORDERKEY = 1 |
| Partition2: | WHERE LO_ORDERKEY = 2 |
| Partition3: | WHERE LO_ORDERKEY = 3 |
| ... |
| PartitionN: | WHERE LO_ORDERKEY = USER_VALUE_N |

You can also create a default partition for values not included in the ones specified.

**To create a LIST virtual partition**

1. Open the AWS SCT application.
2. Choose **Data Migration View** mode.
3. Choose the table where you want to set up virtual partitioning. Open the context (right-click) menu for the table, and choose **Add Virtual Partitioning**.
4. In the **Add Virtual Partitioning** dialog box, enter the information as follows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partition Type</strong></td>
<td>Choose <strong>LIST</strong>. The dialog box UI changes depending on the type you choose.</td>
</tr>
<tr>
<td><strong>Column Name</strong></td>
<td>Choose the column that you want to partition.</td>
</tr>
<tr>
<td><strong>New Value</strong></td>
<td>Type a value here to add it to the set of partitioning values.</td>
</tr>
<tr>
<td><strong>Include Other Values</strong></td>
<td>Choose this option to create a default partition where all values that don't meet the partitioning criteria are stored.</td>
</tr>
<tr>
<td><strong>Load From File</strong></td>
<td>(Optional) Type the name of a .csv file that contains the partition values.</td>
</tr>
</tbody>
</table>
5. Choose **OK**.

**DATE AUTO SPLIT Partition Type**

The DATE AUTO SPLIT partition type partitions data of date and time data types based on a specified interval between a given start date and end date. You specify the data range and interval (day, week, month, or year). If you don't specify a start date or end date, these values default to the current date.

For example, you can create multiple partitions based on a date range you provide. In the following example, the partitioning value range for LO_ORDERDATE is specified to create multiple partitions.

| 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' |

**To create a DATE AUTO SPLIT virtual partition**

1. Open the AWS SCT application.
2. Choose **Data Migration View** mode.
3. Choose the table where you want to set up virtual partitioning. Open the context (right-click) menu for the table, and choose **Add Virtual Partitioning**.
4. In the **Add Virtual Partitioning** dialog box, enter information as follows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partition Type</strong></td>
<td>Choose DATE AUTO SPLIT. The dialog box UI changes depending on the type you choose.</td>
</tr>
<tr>
<td><strong>Column Name</strong></td>
<td>Choose the column that you want to partition.</td>
</tr>
<tr>
<td><strong>Start Date</strong></td>
<td>Type a start date.</td>
</tr>
<tr>
<td><strong>End Date</strong></td>
<td>Type an end date.</td>
</tr>
<tr>
<td><strong>Interval</strong></td>
<td>Type the interval unit, and choose the value for that unit.</td>
</tr>
</tbody>
</table>

5. Choose **OK**.

### Migrating LOBs to Amazon Redshift

Amazon Redshift doesn't support storing large binary objects (LOBs). However, if you need to migrate one or more LOBs to Amazon Redshift, AWS SCT can perform the migration. To do so, AWS SCT uses an Amazon S3 bucket to store the LOBs and writes the URL for the S3 bucket into the migrated data stored in Amazon Redshift.

**To migrate LOBs to Amazon Redshift**

1. Open an AWS SCT project.
2. For **Actions**, choose **Create Local Task**.
3. Choose the **Advanced** tab.
4. For **S3 bucket LOBs folder**, type the name of the folder in an S3 bucket where you want the LOBs stored.
Best Practices and Troubleshooting for Data Extraction Agents

The following are some best practices and troubleshooting suggestions for using extraction agents.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Troubleshooting Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance is slow</td>
<td>To improve performance, we recommend the following:</td>
</tr>
<tr>
<td></td>
<td>• Install multiple agents.</td>
</tr>
<tr>
<td></td>
<td>• Install agents on computers close to your data warehouse.</td>
</tr>
<tr>
<td></td>
<td>• Don't run all tables on a single agent task.</td>
</tr>
<tr>
<td>Contention delays</td>
<td>Avoid having too many agents accessing your data warehouse at the same time.</td>
</tr>
<tr>
<td>An agent goes down temporarily</td>
<td>If an agent is down, the status of each of its tasks appears as failed in AWS SCT. If you wait, in some cases the agent can recover. In this case, the status of its tasks updates in AWS SCT.</td>
</tr>
</tbody>
</table>
## Issue

An agent goes down permanently

## Troubleshooting Suggestions

If the computer running an agent goes down permanently, and that agent is running a task, you can substitute a new agent to continue the task. You can substitute a new agent only if the working folder of the original agent was not on the same computer as the original agent. To substitute a new agent, do the following:

- Install an agent on a new computer.
- Configure the new agent with the same settings, including port number and working folder, as the original agent.
- Start the agent. After the agent starts, the task discovers the new available agent and continues running on the new agent.
Converting Application SQL Using the AWS Schema Conversion Tool

When you convert your database schema from one engine to another, you also need to update the SQL code in your applications to interact with the new database engine instead of the old one. You can view, analyze, edit, and save the converted SQL code.

You can use the AWS Schema Conversion Tool (AWS SCT) to convert the SQL code in your C++, C#, Java, or other application code. For an Oracle to PostgreSQL conversion, you can use AWS SCT to convert SQL*Plus code to PSQL.

Overview of Converting Application SQL

To convert the SQL code in your application, you take the following high-level steps:

- **Create an application conversion project** – The application conversion project is a child of the database schema conversion project. Each database schema conversion project can have one or more child application conversion projects. For more information, see Creating Application Conversion Projects in the AWS Schema Conversion Tool (p. 134).

- **Analyze and convert your SQL code** – AWS SCT analyzes your application, extracts the SQL code, and creates a local version of the converted SQL for you to review and edit. The tool doesn't change the code in your application until you are ready. For more information, see Analyzing and Converting Your SQL Code by Using the AWS Schema Conversion Tool (p. 137).

- **Create an application assessment report** – The application assessment report provides important information about the conversion of the application SQL code from your source database schema to your target database schema. For more information, see Creating and Using the Assessment Report (p. 138).

- **Edit, apply changes to, and save your converted SQL code** – The assessment report includes a list of SQL code items that can't be converted automatically. For these items, you can edit the SQL code manually to perform the conversion. For more information, see Editing and Saving Your Converted SQL Code with the AWS Schema Conversion Tool (p. 139).

Creating Application Conversion Projects in the AWS Schema Conversion Tool

In the AWS Schema Conversion Tool (AWS SCT), the application conversion project is a child of the database schema conversion project. Each database schema conversion project can have one or more child application conversion projects. Use the following procedure to create an application conversion project.

To create an application conversion project

1. In the AWS Schema Conversion Tool, choose **New Application** from the **Applications** menu.
The New application conversion project dialog box appears.

2. Add the following project information.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Project Name</td>
<td>Type a name for your application conversion project. Each database schema conversion project can have one or more child application conversion projects, so choose a name that makes sense if you add more projects later.</td>
</tr>
<tr>
<td>Location</td>
<td>Type the location of the source code for your application.</td>
</tr>
<tr>
<td>Project language</td>
<td>Choose one of the following:</td>
</tr>
<tr>
<td></td>
<td>• JAVA</td>
</tr>
<tr>
<td></td>
<td>• C++</td>
</tr>
</tbody>
</table>
### For This Parameter | Do This
--- | ---
| C# | Any

#### SQL parameter style for target database
Choose one of the following:
- Same as in source
- Positional (?)
- Indexed (:1)
- Indexed ($1)
- Named (@name)
- Named (:name)

#### Select source database schema
In the source tree, choose the schema used by your application code.

3. Choose **OK** to create your application conversion project.

The project window opens.

4. The first time you create an application conversion project, the project window opens automatically. To open an existing application conversion project, select the project node in the source tree, open the context (right-click) menu, and then choose **Manage application**.
5. You can add additional application conversion projects by choosing **New Application** from the **Applications** menu, or by selecting the **Applications** node in the source tree, opening the context (right-click) menu, and then choosing **Add application**.

Analyzing and Converting Your SQL Code by Using the AWS Schema Conversion Tool

Use the following procedure to analyze and convert your SQL code by using the AWS Schema Conversion Tool (AWS SCT).

**To analyze and convert your SQL code**

1. In the application conversion project, choose **Analyze**.

AWS SCT analyzes your application code and extracts the SQL code. A list of all extracted SQL code appears in the **Parsed SQL Scripts** pane at the bottom of the window. The selected item in the list also appears in the **Extracted SQL script** pane.

2. You can analyze each SQL code item in the list, and when you are ready, choose **Convert** to convert the SQL to SQL for your target database.

**Note**

You can edit the converted SQL code in a later procedure.
Creating and Using the Assessment Report

The application assessment report provides important information about converting the application SQL code from your source database schema to your target database schema. The report details all of the SQL extracted from the application, all of the SQL converted, and action items for SQL that can't be converted. The report also includes estimates of the amount of effort that it will take to manually convert the SQL code that can't be converted automatically.

Creating an Application Assessment Report

Use the following procedure to create an application assessment report.

To create an application assessment report

1. In the application conversion project window, choose Create Report from the Actions menu.

   The report is created and opens in the application conversion project window.

2. Review the Summary tab.

   The Summary tab shown following, displays the summary information from the application assessment report. It shows the SQL code items that were converted automatically, and items that were not converted automatically.

3. Choose the SQL Conversion Actions tab and review the information.
The SQL Conversion Actions tab contains a list of SQL code items that can't be converted automatically. There are also recommendations for how to manually convert the SQL code. You edit your converted SQL code in a later step. For more information, see Editing and Saving Your Converted SQL Code with the AWS Schema Conversion Tool (p. 139).

4. You can save a local copy of the application assessment report as either a PDF file or a comma-separated values (CSV) file. The PDF file contains both the summary and action item information. The CSV file contains only action item information.

Editing and Saving Your Converted SQL Code with the AWS Schema Conversion Tool

The assessment report includes a list of SQL code items that can't be converted automatically. For each item that can't be converted, there is an action item on the SQL Conversion Actions tab. For these items, you can edit the SQL code manually to perform the conversion.

Use the following procedure to edit your converted SQL code, apply the changes, and then save them.

To edit, apply changes to, and save your converted SQL code

1. Edit your converted SQL code directly in the Target SQL script pane. If there is no converted code shown, you can click in the pane and start typing.
2. After you are finished editing your converted SQL code, choose Apply. At this point, the changes are saved in memory, but not yet written to your file.
3. Choose Save to save your changes to your file.

Important
When you choose Save you overwrite your original file. Make a copy of your original file before saving so you have a record of your original application code.
Using the AWS Schema Conversion Tool Extension Pack

The AWS SCT Extension Pack is an add-on module that emulates functions present in the source database that are required when converting objects to the target database. Before you can install the AWS SCT Extension Pack, you need to convert your database schema.

The AWS SCT Extension Pack includes the following components:

- **DB schema** – Includes SQL functions, procedures, and tables for emulating some OLTP and OLAP objects (for example, sequence) or unsupported built-in-functions from the source database. This schema is named in the format `aws_<database engine name>_ext`.
- **Custom Python library** (for select OLAP databases) – Includes a set of Python functions that emulate unsupported built-in database functions. Use this library when you migrate from one of the supported databases to Amazon Redshift.
  
  For more information on this library, see Using the Custom Python Library for the AWS SCT Extension Pack (p. 141).
- **AWS Lambda functions** (for select OLTP databases) – Includes AWS Lambda functions that emulate complex database functionality, such as job scheduling and sending emails.

The following sections discuss the AWS SCT Extension Pack.

**Topics**

- Using the Extension Pack Schema (p. 140)
- Using the Custom Python Library for the AWS SCT Extension Pack (p. 141)
- Using the AWS Lambda Functions from the AWS SCT Extension Pack (p. 142)

You can apply the AWS SCT Extension Pack in two ways:

- AWS SCT automatically applies the extension pack when you apply a target database script by choosing **ApplyToTarget** from the context menu. AWS SCT applies the extension pack before it applies all other schema objects.
- To manually apply the extension pack, choose the target database and then choose **Apply Extension Pack** from the context menu. For most situations, automatic application is sufficient. However, you might want to apply the pack if it's accidentally deleted.

Each time the AWS SCT Extension Pack is applied to a target data store, the components are overwritten. Each component has a version number, and AWS SCT warns you if the current component version is older than the one being applied. You can control these notifications in the **Notification Settings** in the Global Settings section of Settings.

Using the Extension Pack Schema

When you convert your database or data warehouse schema, the AWS Schema Conversion Tool (AWS SCT) adds an additional schema to your target database. This schema implements SQL system functions...
of the source database that are required when writing your converted schema to your target database. This additional schema is called the extension pack schema.

The extension pack schema for OLTP databases is named according to the source database as follows:

- Microsoft SQL Server: `AWS_SQLSERVER_EXT`
- MySQL: `AWS_MYSQL_EXT`
- Oracle: `AWS_ORACLE_EXT`
- PostgreSQL: `AWS_POSTGRESQL_EXT`

The extension pack schema for OLAP data warehouse applications is named according to the source data store as follows:

- 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`

### Using the Custom Python Library for the AWS SCT Extension Pack

In some cases, AWS Schema Conversion Tool (AWS SCT) can't convert source database features to equivalent Amazon Redshift features. The AWS SCT Extension Pack contains a custom Python library that emulates some source database functionality on Amazon Redshift.

If you are converting a transactional database, instead see Using the AWS Lambda Functions from the AWS SCT Extension Pack (p. 142).

In two cases, you might want to install the extension pack manually:

- You accidentally delete the extension pack schema from your target database.
- You want to upload custom Python libraries to emulate database functionality.

### Using AWS Services to Upload Custom Python Libraries

The AWS SCT extension pack wizard helps you install the custom Python library.

### Applying the Extension Pack

Use the following procedure to apply the extension pack.

**To apply the extension pack**

1. In the AWS Schema Conversion Tool, in the target database tree, open the context (right-click) menu, and choose **Apply Extension Pack**.
The extension pack wizard appears.

2. Read the Welcome page, and then choose Next.

3. On the AWS Services Settings page, do the following:
   - If you are reinstalling the extension pack schema only, choose Skip this step for now, and then choose Next.
   - If you are uploading the Python library, provide the credentials to connect to your AWS account. You can use your AWS Command Line Interface (AWS CLI) credentials if you have the AWS CLI installed. You can also use credentials that you previously stored in a profile in the global application settings and associated with the project. If necessary, choose Navigate to Project Settings to associate a different profile with the project. If necessary, choose Global Settings to create a new profile. For more information, see Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13).

4. On the Python Library Upload page, do the following:
   - If you are reinstalling the extension pack schema only, choose Skip this step for now, and then choose Next.
   - If you are uploading the Python library, provide the Amazon S3 path, and then choose Upload Library to S3.

When you are done, choose Next.


When you are done, choose Finish.

Using the AWS Lambda Functions from the AWS SCT Extension Pack

The AWSSchema Conversion Tool (AWS SCT) extension pack contains Lambda functions that provide email, job scheduling, and other features to databases hosted on the Amazon EC2 platform.

Using AWS Lambda Functions to Emulate Database Functionality

In some cases, database features can't be converted to equivalent Amazon RDS features. For example, Oracle sends email calls that use UTL_SMTP, and Microsoft SQL Server can use a job scheduler. If you
host and self-manage a database on Amazon EC2, you can emulate these features by substituting AWS services for them.

The AWS SCT extension pack wizard helps you install, create, and configure Lambda functions to emulate email, job scheduling, and other features.

### Applying the Extension Pack

Use the following procedure to apply the extension pack.

**Important**

The AWS service emulation features are supported only for databases installed and self-managed on Amazon EC2. Don't install the service emulation features if your target database is on an Amazon RDS DB instance.

**To apply the extension pack**

1. In the AWS Schema Conversion Tool, in the target database tree, open the context (right-click) menu, and choose **Apply Extension Pack**.

   The extension pack wizard appears.

2. Read the **Welcome** page, and then choose **Next**.

3. On the **AWS Services Settings** page, do the following:

   - If you are reinstalling the extension pack schema only, choose **Skip this step for now**, and then choose **Next**.
   - If you are installing AWS services, provide the credentials to connect to your AWS account. You can use your AWS Command Line Interface (AWS CLI) credentials if you have the AWS CLI installed. You can also use credentials that you previously stored in a profile in the global application settings and associated with the project. If necessary, choose **Navigate to Project Settings** to associate a different profile with the project. If necessary, choose **Global Settings** to create a new profile. For more information, see Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13).

4. On the **Email Sending Service** page, do the following:

   - If you are reinstalling the extension pack schema only, choose **Skip this step for now**, and then choose **Next**.
   - If you are installing AWS services and you have an existing Lambda function, you can provide it. Otherwise, the wizard creates it for you. When you are done, choose **Next**.

5. On the **Job Emulation Service** page, do the following:

   - If you are reinstalling the extension pack schema only, choose **Skip this step for now**, and then choose **Next**.
   - If you are installing AWS services and you have an existing Lambda function, you can provide it. Otherwise, the wizard creates it for you. When you are done, choose **Next**.
6. On the **Functions Emulation** page, choose **Create Extension Pack**. Messages appear with the status of the extension pack operations.

   When you are done, choose **Finish**.
Best Practices for the AWS Schema Conversion Tool

Following, you can find information on best practices and options for using the AWS Schema Conversion Tool (AWS SCT).

General Memory Management and Performance Options

You can configure the AWS Schema Conversion Tool with different memory performance settings. Increasing memory speeds up the performance of your conversion but uses more memory resources on your desktop.

To set your memory management option, choose Global Settings from the Settings menu, and choose the Performance and Memory tab. Choose one of the following options:

- **Fast conversion, but large memory consumption** – This option optimizes for speed of the conversion, but might require more memory for the object reference cache.
- **Low memory consumption, but slower conversion** – This option minimizes the amount of memory used, but results in a slower conversion. Use this option if your desktop has a limited amount of memory.
- **Balance speed with memory consumption** – This option optimizes provides a balance between memory use and conversion speed.

Configuring Additional Memory

For converting large database schemas, for example a database with 3,500 stored procedures, you can configure the amount of memory available to the AWS Schema Conversion Tool.

To modify the amount of memory AWS SCT consumes:

1. Locate the folder where the configuration file is (C:\Program Files\AWS Schema Conversion Tool \App).
2. Open the configuration file AWS Schema Conversion Tool.cfg with Notepad or your favorite text editor.
3. Edit the JVMUserOptions section to set the minimum and maximum memory available. The following example sets the minimum to 4 GB and the maximum to 40 GB.

```
[JVMUserOptions]
-Xmx48960m
-Xms4096m
```
Troubleshooting Issues with the AWS Schema Conversion Tool

Following, you can find information about troubleshooting issues with the AWS Schema Conversion Tool (AWS SCT).

Cannot load objects from an Oracle source database

When you attempt to load schema from an Oracle database, you might encounter one of the following errors.

<table>
<thead>
<tr>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot load objects tree.</td>
</tr>
<tr>
<td>ORA-00942: table or view does not exist</td>
</tr>
</tbody>
</table>

These errors occur because the user whose ID you used to connect to the Oracle database doesn't have sufficient permissions to read the schema, as required by AWS SCT.

You can resolve the issue by granting the user `select_catalog_role` permission and also permission to any dictionary in the database. These permissions provide the read-only access to the views and system tables that is required by AWS SCT. The following example creates a user ID named `min_privs` and grants the user with this ID the minimum permissions required to convert schema from an Oracle source database.

```
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;
```
Release Notes for the AWS Schema Conversion Tool

Release Notes for the AWS Schema Conversion Tool Build 615

The following table shows the features and bug fixes for the AWS Schema Conversion Tool (AWS SCT) version 1.0.615.

<table>
<thead>
<tr>
<th>New feature or enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server to MySQL - Support for MERGE statement</td>
<td>MySQL does not support the MERGE statement, but AWS SCT can emulate the statement by using the INSERT ON DUPLICATE KEY clause and the UPDATE FROM and DELETE FROM statements. For more information, see Converting a SQL Server Schema to MySQL (p. 42).</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - Support for creating unique index names</td>
<td>AWS SCT gives you the option of generating unique index names if your index names are not unique. To do this, choose the option Generate unique index names in the project properties. For more information, see Converting a SQL Server to PostgreSQL (p. 43).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL - Prevent overlapping database sequence values</td>
<td>For Oracle to PostgreSQL migration projects, choose the option Populate converted sequences with the last values generated on the source side in the Conversion settings tab of Project Settings. For more information, see Converting an Oracle Database to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL) (p. 33).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL - Support for PostgreSQL 10 partitioning</td>
<td>AWS SCT can emulate partitions and subpartitions when converting a schema from an Oracle database to a PostgreSQL database. For more information, see Converting an Oracle Database to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL) (p. 33).</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - Support for PostgreSQL 10 partitioning</td>
<td>AWS SCT can emulate partitions and subpartitions when converting a schema from an SQL Server database to a PostgreSQL database. For more information, see Converting a SQL Server to PostgreSQL (p. 43).</td>
</tr>
<tr>
<td>SQL Server to MySQL - Support for GOTO statement</td>
<td>MySQL does not use a GOTO statement. When AWS SCT converts code that contains the GOTO statement, it converts the statement to use a BEGIN...END or LOOP...END LOOP statement.</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - Support for GOTO statement</td>
<td>PostgreSQL does not use a GOTO statement. When AWS SCT converts code that contains the GOTO statement, it converts the statement to use a BEGIN...END or LOOP...END LOOP statement.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL - Support for GOTO statement</td>
<td>PostgreSQL does not use a GOTO statement. When AWS SCT converts code that contains the GOTO statement, it converts the statement to use a BEGIN...END or LOOP...END LOOP statement.</td>
</tr>
</tbody>
</table>
## New feature or enhancement

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 LUW to PostgreSQL - Support for triggers from DB2 to PostgreSQL</td>
<td>AWS SCT can convert the various TRIGGER statements used with DB2 LUW. For more information, see <em>Converting an DB2 LUW Database to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL)</em> (p. 53).</td>
</tr>
<tr>
<td>SQL Server to Amazon RDS for SQL Server - Support for database level triggers</td>
<td>AWS SCT can add database triggers to the object tree when Amazon RDS for SQL Server is the target.</td>
</tr>
<tr>
<td>SQL Server to Amazon RDS for SQL Server - Support for server-level triggers, linked servers, and SQL Server Agents</td>
<td>AWS SCT now supports server-level triggers, linked servers, and SQL Server Agents. For more information, see <em>Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT)</em> (p. 37).</td>
</tr>
<tr>
<td>Oracle to Amazon RDS for Oracle - Support for directory objects, tablespaces, and user roles and privileges</td>
<td>AWS SCT can add directory objects to the object tree. For more information, see <em>Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT)</em> (p. 37).</td>
</tr>
<tr>
<td>Db2 support version 10.1</td>
<td>AWS SCT now supports IBM Db2 LUW version 10.1.</td>
</tr>
<tr>
<td>Added support of isolated government regions</td>
<td>AWS SCT now supports isolated government regions.</td>
</tr>
</tbody>
</table>

## Issues Resolved

- AWS Profile. Working with federated credentials.
- DB2 to PostgreSQL. Conversion of DB2 triggers.
- Assessment Report. Flag LOB tables without Primary Key.
- Some UI bugs were fixed.
Release Notes for the AWS Schema Conversion Tool Build 614

The following table shows the features and bug fixes for the AWS Schema Conversion Tool (AWS SCT) version 1.0.614.

<table>
<thead>
<tr>
<th>New feature or enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle to PostgreSQL: Assessment Report for SQL*Plus Conversion</td>
<td>AWS SCT can convert SQL<em>Plus files into PSQL. The assessment report shows how AWS SCT converted the SQL</em>Plus files into PSQL. For more information, see Using Oracle as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 29).</td>
</tr>
<tr>
<td>Aurora MySQL Compatible 5.7 version support</td>
<td>Added support for converting Aurora MySQL 5.7 schemas.</td>
</tr>
<tr>
<td>Db2 LUW 9.1</td>
<td>Added support for Db2 LUW version 9.1.</td>
</tr>
<tr>
<td>OLTP Data Migration. Data compression.</td>
<td>Data compression during migration is now optional when you set up a task using a Replication agent.</td>
</tr>
<tr>
<td>Oracle to Oracle RDS: DB Links support</td>
<td>Oracle to Oracle RDS migrations now support DB Links. For more information, see Using Oracle as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 29).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL: SELECT INTO BULK COLLECT (VARRAY) conversion</td>
<td>SQL statements using BULK COLLECT (VARRAY) can now be converted when migrating between Oracle and PostgreSQL.</td>
</tr>
<tr>
<td>Oracle comments conversion</td>
<td>Added support for converting Oracle comments into the format used by the target database engine. For more information, see Using Oracle as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 29).</td>
</tr>
<tr>
<td>Emulation of Oracle system objects</td>
<td>Added support for converting Oracle system objects into PostgreSQL. For more information, see Using Oracle as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 29).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL: ROWNUM converted to LIMIT</td>
<td>Added support for converting ROWNUM.</td>
</tr>
<tr>
<td>Microsoft SQL Server to Microsoft SQL Server RDS: BULK INSERT and OPENROWSET()</td>
<td>Added support converting BULK INSERT and OPENROWSET()</td>
</tr>
<tr>
<td>Microsoft SQL Server to Microsoft SQL Server RDS: Links inside storage objects</td>
<td>AWS SCT now supports links inside stored objects during a migration to Amazon RDS. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 37).</td>
</tr>
<tr>
<td>Microsoft SQL Server to PostgreSQL: PATINDEX</td>
<td>AWS SCT now supports converting PATINDEX during a migration to PostgreSQL. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 37).</td>
</tr>
<tr>
<td>Microsoft SQL Server to PostgreSQL: System objects access</td>
<td>SQL Server system object are now converted to objects in PostgreSQL. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT) (p. 37).</td>
</tr>
</tbody>
</table>
### New feature or enhancement

<table>
<thead>
<tr>
<th>New feature or enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server to PostgreSQL: Inline function creation</td>
<td>Added support for inline functions. For more information, see [Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (AWS SCT)](p. 37).</td>
</tr>
</tbody>
</table>

### Issues Resolved

- Type Mapping. Bug fixing and improvements.
- Oracle to PostgreSQL. Dynamic SQL Conversion bug fixing and improvements
- SQL*Plus script conversion. Bug fixing and improvements
- Oracle to PostgreSQL Conversion. Bind variables recognition fix.
- Added ability to update server info by clicking "Update server info" on server level.
- Netezza. Schema conversion for objects in lower-case fixed.
- Handling some specific characters in the AWS SCT navigation tree nodes' names derived from file names
- Some UI bugs were fixed.
# Release Notes for the AWS Schema Conversion Tool Build 613

The following table shows the features and bug fixes for the AWS Schema Conversion Tool (AWS SCT) version 1.0.613.

<table>
<thead>
<tr>
<th>New feature or enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 LUW support</td>
<td>Added support for migrating a DB2 database as a source. For more information, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 70).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion. Added SQL*Plus files conversion</td>
<td>You can now convert SQL*Plus files.</td>
</tr>
<tr>
<td>Snowball Tab added for OLTP Data Migration</td>
<td>A new Snowball tab was added for OLTP Data Migration that shows you the current status of a Snowball device for particular project. For more information, see Using Data Extraction Agents (p. 110).</td>
</tr>
<tr>
<td>Oracle To PostgreSQL Conversion. Convert Oracle spatial code to PostGIS</td>
<td>Oracle To PostgreSQL Conversion. Oracle spatial code converts to PostGIS.</td>
</tr>
<tr>
<td>Schema Compare Added Oracle 10 Support</td>
<td>You can now run Schema Compare with Oracle 10. For more information, see Comparing Database Schemas (p. 86)</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion. Implicit typecasting of Oracle db handled in PostgreSQL</td>
<td>While converting from Oracle to PostgreSQL, SCT add datatype casting.</td>
</tr>
<tr>
<td>SQL Server To PostgreSQL Conversion</td>
<td>Added CITEXT type support. Now you can choose it with data type mappint.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion</td>
<td>Additional improvements of Dynamic SQL Conversion for EXECUTE IMMEDIATE and DBMS_SQL and Cursors.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion</td>
<td>Added support of SELECT INTO BULK COLLECT Conversion for Oracle to PostgreSQL.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion</td>
<td>Now it is possible to convert an Associative array from Oracle to PostgreSQL.</td>
</tr>
<tr>
<td>Type Mapping - Custom type mapping improvements</td>
<td>Added ability to select source data types based on length and precision.</td>
</tr>
<tr>
<td>AWS Profile Settings</td>
<td>Added ability to select default profile in AWS Profile Settings.</td>
</tr>
<tr>
<td>Greenplum to Redshift Conversion</td>
<td>BuiltIn SQL Functions to Redshift Scalar SQL UDF conversion added.</td>
</tr>
</tbody>
</table>
## Issues Resolved

<table>
<thead>
<tr>
<th>Date reported</th>
<th>Description</th>
</tr>
</thead>
</table>
| post version 612 | • Oracle to Redshift Conversion. CAST to CHAR working, CAST to CHARACTER does not work.  
• Comments on tables aren't created in the DDL of the source Oracle DB.  
• Type mismatch isn't flagged in assessment report.  
• Oracle to PostgreSQL Conversion. Wrongly converts full-width uppercase alphabet table name to full-width lowercase alphabet table name.  
• Global Settings Window cannot be closed by clicking the OK button.  
• Oracle to Redshift. to_date and subtraction re-write gives different result.  
• Some UI bugs were fixed. |
Release Notes for the AWS Schema Conversion Tool Build 612

The following table shows the features and bug fixes for the AWS Schema Conversion Tool (AWS SCT) version 1.0.612.

<table>
<thead>
<tr>
<th>New feature or enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Data Type Mapping</td>
<td>You can now set up mapping rules to change the data type for storage objects. You define what data type should be changed in Mapping Rules tab.</td>
</tr>
<tr>
<td>What's New option added</td>
<td>A new &quot;What's New&quot; option under the Help menu shows you all major features that were added for this release.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion.</td>
<td>The migration from Oracle to PostgreSQL now supports</td>
</tr>
<tr>
<td></td>
<td>• global variables</td>
</tr>
<tr>
<td></td>
<td>• associative arrays</td>
</tr>
<tr>
<td></td>
<td>• formatting strings for TO_NUMBER function</td>
</tr>
<tr>
<td></td>
<td>• converting Dynamic SQL with DMS_SQL package</td>
</tr>
<tr>
<td></td>
<td>• converting multiple nested subprograms with global variables</td>
</tr>
<tr>
<td>Added support for loading some PostgreSQL function attributes and domain constraints</td>
<td>Added support for loading PostgreSQL IMMUTABLE, STABLE, and VOLATILE function attributes.</td>
</tr>
<tr>
<td></td>
<td>Added support for loading PostgreSQL domain constraints.</td>
</tr>
</tbody>
</table>

Issues Resolved

<table>
<thead>
<tr>
<th>Date reported</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>post version 611</td>
<td>• APPLY is recognized as a keyword for Oracle.</td>
</tr>
<tr>
<td></td>
<td>• Fixed error when running Schema Compare for an Oracle-to-Oracle project.</td>
</tr>
<tr>
<td></td>
<td>• General improvements based on feedback and bug fixes for Microsoft SQL Server to Microsoft SQL Server migrations.</td>
</tr>
<tr>
<td></td>
<td>• DROP FOREIGN KEY CONSTRAINTS missing when using Save as SQL for Oracle to PostgreSQL migrations.</td>
</tr>
<tr>
<td></td>
<td>• TO_DATE &amp; TRUNC Function conversion bug fixes for Oracle to Redshift migrations.</td>
</tr>
<tr>
<td></td>
<td>• General improvements based on feedback and bug fixes for PostgreSQL to PostgreSQL migrations.</td>
</tr>
<tr>
<td></td>
<td>• General improvements based on feedback and bug fixes for MySQL to MySQL migrations.</td>
</tr>
<tr>
<td></td>
<td>• General improvements based on feedback and bug fixes for Oracle to Oracle migrations.</td>
</tr>
<tr>
<td></td>
<td>• Some UI bugs were fixed.</td>
</tr>
<tr>
<td>Date reported</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>• Rewrite NUMTOINTERVL() to Redshift interval literal for Oracle to Redshift migrations.</td>
</tr>
<tr>
<td></td>
<td>• Performance optimization.</td>
</tr>
</tbody>
</table>
# Release Notes for the AWS Schema Conversion Tool Build 611

The following table shows the features and bug fixes for the AWS Schema Conversion Tool (AWS SCT) version 1.0.611.

<table>
<thead>
<tr>
<th>New feature or enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Compare for MySQL to MySQL</td>
<td>Added the ability to compare MySQL to MySQL databases. For more information, see <a href="#">Comparing Database Schemas</a>.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL dynamic statements conversion</td>
<td>Added first version of DBMS_SQL package conversion support. For more information, see <a href="#">Converting Dynamic SQL for Oracle to PostgreSQL Migrations</a>.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL GOTO statement conversion</td>
<td>PostgreSQL doesn't support the GOTO operator in functionality such as Oracle, but it can be converted using BEGIN/END or LOOP/END loop statements.</td>
</tr>
<tr>
<td>Open log file from error message</td>
<td>When you encounter an error, you can click on it and have it take you to the associated log file rather than having to search around on the source system for it.</td>
</tr>
<tr>
<td>Added Estimated Complexity field to PDF export of Assessment Report</td>
<td>The Estimated Complexity field is exported in the .pdf version of the Assessment report, but it is not included in the .csv version. For more information, see <a href="#">Creating and Using the Assessment Report in the AWS Schema Conversion Tool</a>.</td>
</tr>
<tr>
<td>OLAP Data Migration. Added option to not delete files on S3 after Redshift copy</td>
<td>After a migration to Amazon Redshift, the agent can either keep or delete the uploaded files. For more information, see <a href="#">Optimizing Amazon Redshift by Using the AWS Schema Conversion Tool</a>.</td>
</tr>
<tr>
<td>OLAP Data Migration. Added LOB migration support for Greenplum, Vertica, Netezza, and Microsoft SQL Server.</td>
<td>Added ability to migrate LOB columns. For more information, see <a href="#">Migrating LOBs to Amazon Redshift</a>.</td>
</tr>
<tr>
<td>Added ability to see related objects for conversions such as Oracle to MySQL or Aurora for MySQL and Microsoft SQL to MySQL or Aurora for MySQL.</td>
<td>When AWS SCT transforms source object into multiple target objects, you can now see a full list of related objects that were created. For more information, see <a href="#">Finding Related Transformed Objects</a>.</td>
</tr>
<tr>
<td>OLAP Data Extractors. Added possibility to recover agent after reinstall</td>
<td>During installation or configuration, you can recover the agent if port or location changed. For more information, see <a href="#">Hiding and Recovering Information for an AWS SCT Agent</a>.</td>
</tr>
<tr>
<td>Ability to hide schemas in tree view</td>
<td>You can decide what objects and information in your schemas you want to view in tree view. For more information, see <a href="#">Hiding Schemas in the AWS SCT Tree View</a>.</td>
</tr>
<tr>
<td>Supports virtual partitioning</td>
<td>You can now manage large non-partitioned tables by creating subtasks that create virtual partitions of the table data using filtering rules. For</td>
</tr>
</tbody>
</table>
### New feature or enhancement

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>more information, see [Using Virtual Partitioning with AWS Schema Conversion Tool](p. 128).</td>
</tr>
</tbody>
</table>

### Issues Resolved

<table>
<thead>
<tr>
<th>Date reported</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>post version 608</td>
<td>• OLTP Data Migration. Added the ability to refresh Agent/Task Logs.</td>
</tr>
<tr>
<td></td>
<td>• Oracle to Oracle schema compare. Fixed the comparison of public synonyms.</td>
</tr>
<tr>
<td></td>
<td>• Fixed the ability to hide system schema from Tree View.</td>
</tr>
<tr>
<td></td>
<td>• Added a Tooltip for the tree filter. Allows user to upload .csv files with objects that needs to be filtered.</td>
</tr>
<tr>
<td></td>
<td>• Fixed conversion of Oracle NUMTOINTERVAL to Redshift interval literal where applicable.</td>
</tr>
<tr>
<td></td>
<td>• Oracle to Redshift Conversion. Fixed the migration of a Redshift SUBSTR with a second parameter constant (start of substr) to be a simple expression rather than a CASE statement.</td>
</tr>
<tr>
<td></td>
<td>• OLAP Data Migration. Added description for virtual partitioning.</td>
</tr>
</tbody>
</table>
### Document History

The following table describes the important changes to the documentation since the last release of the AWS Schema Conversion Tool (AWS SCT) User Guide.

<table>
<thead>
<tr>
<th>Version</th>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.615</td>
<td>Updates and fixes</td>
<td>See the Release Notes for the AWS Schema Conversion Tool (p. 147) section for more information.</td>
<td>May 24, 2018</td>
</tr>
<tr>
<td>1.0.614</td>
<td>Updates and fixes</td>
<td>See the Release Notes for the AWS Schema Conversion Tool (p. 147) section for more information.</td>
<td>April 25, 2018</td>
</tr>
<tr>
<td>1.0.613</td>
<td>Updates and fixes</td>
<td>See the Release Notes for the AWS Schema Conversion Tool (p. 147) section for more information.</td>
<td>March 28, 2018</td>
</tr>
<tr>
<td>1.0.612</td>
<td>Updates and fixes</td>
<td>See the Release Notes for the AWS Schema Conversion Tool (p. 147) section for more information.</td>
<td>February 22, 2018</td>
</tr>
<tr>
<td>1.0.611</td>
<td>Updates and fixes</td>
<td>See the Release Notes for the AWS Schema Conversion Tool (p. 147) section for more information.</td>
<td>January 23, 2018</td>
</tr>
<tr>
<td>1.0.608</td>
<td><strong>FIPS endpoint support for Amazon S3</strong></td>
<td>You can now request AWS SCT to connect to Amazon S3 and Amazon Redshift by using FIPS endpoints to comply with Federal Information Processing Standard security requirements. For more information, see Storing AWS Credentials (p. 14).</td>
<td>November 17, 2017</td>
</tr>
<tr>
<td>1.0.607</td>
<td><strong>FIPS endpoint support for Amazon S3</strong></td>
<td>You can now request AWS SCT to connect to Amazon S3 and Amazon Redshift by using FIPS endpoints to comply with Federal Information Processing Standard security requirements. For more information, see Storing AWS Credentials (p. 14).</td>
<td>October 30, 2017</td>
</tr>
<tr>
<td>1.0.607</td>
<td>Data extraction tasks can ignore LOBs</td>
<td>When you create data extraction tasks, you can now choose to ignore large objects (LOBs) to reduce the amount of data that you extract. For more information, see Creating, Running, and Monitoring an AWS SCT Data Extraction Task (p. 119).</td>
<td>October 30, 2017</td>
</tr>
<tr>
<td>1.0.605</td>
<td>Data extraction agent task log access</td>
<td>You can now access the data extraction agent task log from a convenient link in the AWS Schema Conversion Tool user interface. For more information, see Creating, Running, and Monitoring an AWS SCT Data Extraction Task (p. 119).</td>
<td>August 28, 2017</td>
</tr>
<tr>
<td>Version</td>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1.0.604</td>
<td>Converter enhancements</td>
<td>The AWS Schema Conversion Tool engine has been enhanced to offer improved conversions for heterogeneous migrations.</td>
<td>June 24, 2017</td>
</tr>
<tr>
<td>1.0.603</td>
<td>Data extraction agents support</td>
<td>You can now filter the data that the extraction agents extract from your data warehouse. For more information, see Creating Data Extraction Filters in the AWS Schema Conversion Tool (p. 117).</td>
<td>June 16, 2017</td>
</tr>
<tr>
<td>1.0.603</td>
<td>AWS SCT supports additional data</td>
<td>You can now use the AWS Schema Conversion Tool to convert your Terradata 13 and Oracle Data Warehouse 10 schemas to equivalent Amazon Redshift schemas. For more information, see Converting Data Warehouse Schemas to Amazon Redshift by Using the AWS Schema Conversion Tool (p. 88).</td>
<td>June 16, 2017</td>
</tr>
<tr>
<td>1.0.602</td>
<td>Data extraction agents support</td>
<td>You can now use data extraction agents to extract data from your Microsoft SQL Server data warehouses. For more information, see Using Data Extraction Agents (p. 110).</td>
<td>May 11, 2017</td>
</tr>
<tr>
<td>1.0.602</td>
<td>Data extraction agents can copy</td>
<td>Data extraction agents now have three upload modes. You can now specify whether to just extract your data, to extract your data and just upload it to Amazon S3, or to extract, upload, and copy your data directly into Amazon Redshift. For more information, see Creating, Running, and Monitoring an AWS SCT Data Extraction Task (p. 119).</td>
<td>May 11, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>AWS SCT supports additional data</td>
<td>You can now use the AWS Schema Conversion Tool to convert your Vertica and Microsoft SQL Server schemas to equivalent Amazon Redshift schemas. For more information, see Converting Data Warehouse Schemas to Amazon Redshift by Using the AWS Schema Conversion Tool (p. 88).</td>
<td>April 18, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>Data extraction agents support</td>
<td>You can now use data extraction agents to extract data from your Greenplum, Netezza, and Vertica data warehouses. For more information, see Using Data Extraction Agents (p. 110).</td>
<td>April 18, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>Data extraction agents support</td>
<td>You can now install data extraction agents on computers running the macOS and Microsoft Windows operating systems. For more information, see Installing Extraction Agents (p. 113).</td>
<td>April 18, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>Data extraction agents upload to</td>
<td>Data extraction agents now upload your extracted data to Amazon S3 automatically. For more information, see Data Extraction Task Output (p. 127).</td>
<td>April 18, 2017</td>
</tr>
<tr>
<td></td>
<td>Amazon S3 automatically</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>1.0.600</td>
<td>Data Extraction Agents</td>
<td>You can now install data extraction agents that extract data from your data warehouse and prepare it for use with Amazon Redshift. You can use the AWS Schema Conversion Tool to register the agents and create data extraction tasks for them. For more information, see Using Data Extraction Agents (p. 110).</td>
<td>February 16, 2017</td>
</tr>
<tr>
<td>1.0.600</td>
<td>Customer Feedback</td>
<td>You can now provide feedback about the AWS Schema Conversion Tool. You can file a bug report, you can submit a feature request, or you can provide general information. For more information, see Providing Customer Feedback (p. 2).</td>
<td>February 16, 2017</td>
</tr>
<tr>
<td>1.0.502</td>
<td>Integration with AWS DMS</td>
<td>You can now use the AWS Schema Conversion Tool to create AWS DMS endpoints and tasks. You can run and monitor the tasks from AWS SCT. For more information, see Using the AWS Schema Conversion Tool with the AWS Database Migration Service (p. 109).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>1.0.502</td>
<td>Amazon Aurora with PostgreSQL compatibility as a target database</td>
<td>The AWS Schema Conversion Tool now supports Amazon Aurora with PostgreSQL compatibility as a target database. For more information, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 70).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>1.0.502</td>
<td>Support for profiles</td>
<td>You can now store different profiles in the AWS Schema Conversion Tool and easily switch between them. For more information, see Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>1.0.501</td>
<td>Support for Greenplum Database and Netezza</td>
<td>You can now use the AWS Schema Conversion Tool to convert your data warehouse schemas from Greenplum Database and Netezza to Amazon Redshift. For more information, see Converting Data Warehouse Schemas to Amazon Redshift by Using the AWS Schema Conversion Tool (p. 88).</td>
<td>November 17, 2016</td>
</tr>
<tr>
<td>1.0.501</td>
<td>Redshift optimization</td>
<td>You can now use the AWS Schema Conversion Tool to optimize your Amazon Redshift databases. For more information, see Optimizing Amazon Redshift by Using the AWS Schema Conversion Tool (p. 107).</td>
<td>November 17, 2016</td>
</tr>
<tr>
<td>1.0.500</td>
<td>Mapping rules</td>
<td>Before you convert your schema with the AWS Schema Conversion Tool, you can now set up rules that change the data type of columns, move objects from one schema to another, and change the names of objects. For more information, see Creating Mapping Rules in the AWS Schema Conversion Tool (p. 91).</td>
<td>October 4, 2016</td>
</tr>
<tr>
<td>Version</td>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>1.0.500</td>
<td>Move to cloud</td>
<td>You can now use the AWS Schema Conversion Tool to copy your existing on-premises database schema to an Amazon RDS DB instance running the same engine. You can use this feature to analyze potential cost savings of moving to the cloud and of changing your license type. For more information, see Creating and Using the Assessment Report in the AWS Schema Conversion Tool (p. 77).</td>
<td>October 4, 2016</td>
</tr>
<tr>
<td>1.0.400</td>
<td>Data warehouse schema conversions</td>
<td>You can now use the AWS Schema Conversion Tool to convert your data warehouse schemas from Oracle and Teradata to Amazon Redshift. For more information, see Converting Data Warehouse Schemas to Amazon Redshift by Using the AWS Schema Conversion Tool (p. 88).</td>
<td>July 13, 2016</td>
</tr>
<tr>
<td>1.0.400</td>
<td>Application SQL conversions</td>
<td>You can now use the AWS Schema Conversion Tool to convert SQL in your C++, C#, Java, or other application code. For more information, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 134).</td>
<td>July 13, 2016</td>
</tr>
<tr>
<td>1.0.400</td>
<td>New feature</td>
<td>The AWS Schema Conversion Tool now contains an extension pack and a wizard to help you install, create, and configure AWS Lambda functions and Python libraries to provide email, job scheduling, and other features. For more information, see Using the AWS Lambda Functions from the AWS SCT Extension Pack (p. 142) and Using the Custom Python Library for the AWS SCT Extension Pack (p. 141).</td>
<td>July 13, 2016</td>
</tr>
<tr>
<td>1.0.301</td>
<td>SSL Support</td>
<td>You can now use Secure Sockets Layer (SSL) to connect to your source database when you use the AWS Schema Conversion Tool.</td>
<td>May 19, 2016</td>
</tr>
<tr>
<td>1.0.203</td>
<td>New feature</td>
<td>Adds support for MySQL and PostgreSQL as source databases for conversions.</td>
<td>April 11, 2016</td>
</tr>
<tr>
<td>1.0.202</td>
<td>Maintenance release</td>
<td>Adds support for editing the converted SQL that was generated for the target database engine. Adds improved selection capabilities in the source database and target DB instance tree views. Adds support for connecting to an Oracle source database using Transparent Network Substrate (TNS) names.</td>
<td>March 2, 2016</td>
</tr>
<tr>
<td>1.0.200</td>
<td>Maintenance release</td>
<td>Adds support for PostgreSQL as a target database engine. Adds the ability to generate converted schema as scripts and to save the scripts to files prior to applying the schema to the target DB instance.</td>
<td>January 14, 2016</td>
</tr>
<tr>
<td>Version</td>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>1.0.103</td>
<td>Maintenance release</td>
<td>Adds offline project capability, the ability to check for new versions, and memory and performance management.</td>
<td>December 2, 2015</td>
</tr>
<tr>
<td>1.0.101</td>
<td>Maintenance release</td>
<td>Adds the <strong>Create New Database Migration Project</strong> wizard. Adds the ability to save the database migration assessment report as a PDF file.</td>
<td>October 19, 2015</td>
</tr>
<tr>
<td>1.0.100</td>
<td>Preview release</td>
<td>Provides the user guide for the AWS Schema Conversion Tool preview release.</td>
<td>October 7, 2015</td>
</tr>
</tbody>
</table>