<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is AWS SCT? ..................</td>
</tr>
<tr>
<td>Schema conversion overview ..................</td>
</tr>
<tr>
<td>Giving feedback ..................</td>
</tr>
<tr>
<td>Installing, verifying, and updating ..................</td>
</tr>
<tr>
<td>Installing AWS SCT ..................</td>
</tr>
<tr>
<td>Installing previous versions ..................</td>
</tr>
<tr>
<td>Verifying the AWS SCT file download ..................</td>
</tr>
<tr>
<td>Verifying the checksum of the AWS SCT file ..................</td>
</tr>
<tr>
<td>Verifying the AWS SCT RPM files on Fedora ..................</td>
</tr>
<tr>
<td>Verifying the AWS SCT DEB files on Ubuntu ..................</td>
</tr>
<tr>
<td>Verifying the AWS SCT MSI file on Microsoft Windows ..................</td>
</tr>
<tr>
<td>Installing the required database drivers ..................</td>
</tr>
<tr>
<td>Installing JDBC drivers on Linux ..................</td>
</tr>
<tr>
<td>Storing driver paths in the global settings ..................</td>
</tr>
<tr>
<td>Updating AWS SCT ..................</td>
</tr>
<tr>
<td>Using the AWS SCT user interface ..................</td>
</tr>
<tr>
<td>The project window ..................</td>
</tr>
<tr>
<td>Starting AWS SCT ..................</td>
</tr>
<tr>
<td>Creating a project ..................</td>
</tr>
<tr>
<td>Saving and opening a project ..................</td>
</tr>
<tr>
<td>Adding a server ..................</td>
</tr>
<tr>
<td>Using an offline mode ..................</td>
</tr>
<tr>
<td>Using tree filters ..................</td>
</tr>
<tr>
<td>Importing a file list for the tree filter ..................</td>
</tr>
<tr>
<td>Hiding schemas ..................</td>
</tr>
<tr>
<td>Managing the database migration assessment report ..................</td>
</tr>
<tr>
<td>Converting your schema ..................</td>
</tr>
<tr>
<td>Applying the converted schema to your target DB instance ..................</td>
</tr>
<tr>
<td>Storing AWS profiles ..................</td>
</tr>
<tr>
<td>Storing AWS credentials ..................</td>
</tr>
<tr>
<td>Setting the default profile for a project ..................</td>
</tr>
<tr>
<td>Using AWS Secrets Manager ..................</td>
</tr>
<tr>
<td>Storing database passwords ..................</td>
</tr>
<tr>
<td>Using the Union All view for projects with partitioned tables ..................</td>
</tr>
<tr>
<td>Keyboard shortcuts ..................</td>
</tr>
<tr>
<td>Getting started ..................</td>
</tr>
<tr>
<td>Sources for AWS SCT ..................</td>
</tr>
<tr>
<td>Encrypting Amazon RDS connections ..................</td>
</tr>
<tr>
<td>Using Apache Cassandra as a source ..................</td>
</tr>
<tr>
<td>Connecting to Apache Cassandra as a source ..................</td>
</tr>
<tr>
<td>Using Azure SQL Database as a source ..................</td>
</tr>
<tr>
<td>Privileges for Azure SQL Database ..................</td>
</tr>
<tr>
<td>Connecting to Azure SQL Database as a source ..................</td>
</tr>
<tr>
<td>Using IBM Db2 for z/OS as a source ..................</td>
</tr>
<tr>
<td>Privileges for Db2 for z/OS ..................</td>
</tr>
<tr>
<td>Connecting to Db2 for z/OS as a source ..................</td>
</tr>
<tr>
<td>Using IBM Db2 LUW as a source ..................</td>
</tr>
<tr>
<td>Privileges for Db2 LUW ..................</td>
</tr>
<tr>
<td>Connecting to Db2 LUW as a source ..................</td>
</tr>
<tr>
<td>Db2 LUW to PostgreSQL ..................</td>
</tr>
<tr>
<td>Using MySQL as a source ..................</td>
</tr>
<tr>
<td>Privileges for MySQL ..................</td>
</tr>
<tr>
<td>Connecting to MySQL as a source ..................</td>
</tr>
</tbody>
</table>
Using Oracle Database as a source .................................................................................. 57
  Privileges for Oracle ........................................................................................................ 57
  Connecting to Oracle as a source .................................................................................... 57
  Oracle to PostgreSQL ...................................................................................................... 59
  Oracle to MySQL .............................................................................................................. 62
  Oracle to Amazon RDS for Oracle .................................................................................... 66
Using PostgreSQL as a source ........................................................................................ 69
  Privileges for PostgreSQL ............................................................................................... 69
  Connecting to PostgreSQL as a source .......................................................................... 69
Using SQL Server as a source .......................................................................................... 71
  Privileges for Microsoft SQL Server ............................................................................. 71
  Using Windows Authentication with Microsoft SQL Server .......................................... 71
  Connecting to SQL Server as a source .......................................................................... 73
  SQL Server to MySQL ..................................................................................................... 74
  SQL Server to PostgreSQL ................................................................................................ 75
  SQL Server to Amazon RDS SQL Server ........................................................................ 98
Using SAP ASE (Sybase ASE) as a source ........................................................................ 98
  Privileges for SAP ASE ................................................................................................. 99
  Connecting to SAP ASE as a source ............................................................................. 99
Data warehouse sources for AWS SCT ............................................................................. 101
  Using Amazon Redshift as a source ............................................................................. 101
  Using Azure Synapse Analytics as a source ................................................................. 103
  Using Greenplum Database as a source ........................................................................ 104
  Using Netezza as a source ............................................................................................. 106
  Using Oracle Data Warehouse as a source ................................................................... 109
  Using Snowflake as a source .......................................................................................... 111
  Using SQL Server Data Warehouse as a source ............................................................ 115
  Using Teradata as a source ............................................................................................ 117
  Using Vertica as a source .............................................................................................. 119
Creating mapping rules .................................................................................................... 122
  New rule .......................................................................................................................... 122
  Managing rules .............................................................................................................. 122
  Virtual targets ................................................................................................................. 123
  Limitations ........................................................................................................................ 124
Creating conversion reports ........................................................................................... 125
  Migration assessment reports ......................................................................................... 125
    Creating a database migration assessment report ......................................................... 126
    Viewing the assessment report .................................................................................... 126
    Saving the assessment report ...................................................................................... 133
    Creating a multiserver assessment report .................................................................. 136
Converting database schemas ......................................................................................... 142
  Creating migration rules ................................................................................................. 143
    Creating migration rules ............................................................................................... 144
    Exporting migration rules ............................................................................................. 146
  Converting your schema ................................................................................................. 146
    Converting schema ........................................................................................................ 146
    Editing converted schema ............................................................................................. 150
    Clearing a converted schema ......................................................................................... 151
  Handling manual conversions ........................................................................................ 152
    Modifying your source schema ..................................................................................... 152
    Modifying your target schema ........................................................................................ 152
  Updating and refreshing your converted schema ............................................................ 152
  Saving and applying your schema .................................................................................. 153
    Saving your converted schema ..................................................................................... 153
    Applying your converted schema ................................................................................. 154
    The extension pack schema .......................................................................................... 154
  Comparing schemas ....................................................................................................... 155
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post-migration activities</td>
<td>267</td>
</tr>
<tr>
<td>Converting application SQL</td>
<td>268</td>
</tr>
<tr>
<td>Overview of converting application SQL</td>
<td>268</td>
</tr>
<tr>
<td>Converting SQL code in your applications</td>
<td>268</td>
</tr>
<tr>
<td>Creating generic application conversion projects</td>
<td>269</td>
</tr>
<tr>
<td>Managing application conversion projects</td>
<td>272</td>
</tr>
<tr>
<td>Analyzing and converting your SQL code</td>
<td>272</td>
</tr>
<tr>
<td>Creating and using the assessment report</td>
<td>273</td>
</tr>
<tr>
<td>Editing and saving your converted SQL code</td>
<td>274</td>
</tr>
<tr>
<td>Converting SQL code in C# applications</td>
<td>275</td>
</tr>
<tr>
<td>Creating C# application conversion projects</td>
<td>275</td>
</tr>
<tr>
<td>Managing C# application conversion projects</td>
<td>276</td>
</tr>
<tr>
<td>Converting your C# application SQL code</td>
<td>277</td>
</tr>
<tr>
<td>Creating a C# application conversion assessment report</td>
<td>278</td>
</tr>
<tr>
<td>Saving your converted application code</td>
<td>279</td>
</tr>
<tr>
<td>Converting SQL code in C++ applications</td>
<td>279</td>
</tr>
<tr>
<td>Creating C++ application conversion projects</td>
<td>279</td>
</tr>
<tr>
<td>Converting your C++ application SQL code</td>
<td>280</td>
</tr>
<tr>
<td>Creating a C++ application conversion assessment report</td>
<td>284</td>
</tr>
<tr>
<td>Saving your converted application code</td>
<td>284</td>
</tr>
<tr>
<td>Converting SQL code in Java applications</td>
<td>285</td>
</tr>
<tr>
<td>Creating Java application conversion projects</td>
<td>285</td>
</tr>
<tr>
<td>Managing Java application conversion projects</td>
<td>286</td>
</tr>
<tr>
<td>Converting your Java application SQL code</td>
<td>287</td>
</tr>
<tr>
<td>Creating a Java application conversion assessment report</td>
<td>288</td>
</tr>
<tr>
<td>Saving your converted application code</td>
<td>289</td>
</tr>
<tr>
<td>Converting SQL code in Pro*C applications</td>
<td>290</td>
</tr>
<tr>
<td>Creating Pro*C application conversion projects</td>
<td>290</td>
</tr>
<tr>
<td>Managing Pro*C application conversion projects</td>
<td>291</td>
</tr>
<tr>
<td>Converting your Pro*C application SQL code</td>
<td>292</td>
</tr>
<tr>
<td>Creating a Pro*C application conversion assessment report</td>
<td>293</td>
</tr>
<tr>
<td>Editing and saving your converted application code</td>
<td>294</td>
</tr>
<tr>
<td>Using extension packs</td>
<td>296</td>
</tr>
<tr>
<td>Using the extension pack schema</td>
<td>296</td>
</tr>
<tr>
<td>Custom Python library for extension packs</td>
<td>297</td>
</tr>
<tr>
<td>Using AWS services to upload the custom Python library</td>
<td>297</td>
</tr>
<tr>
<td>Applying the extension pack</td>
<td>297</td>
</tr>
<tr>
<td>Using the Lambda functions from the AWS SCT extension pack</td>
<td>298</td>
</tr>
<tr>
<td>Using AWS Lambda functions to emulate database functionality</td>
<td>298</td>
</tr>
<tr>
<td>Applying the extension pack to support Lambda functions</td>
<td>299</td>
</tr>
<tr>
<td>Best practices</td>
<td>301</td>
</tr>
<tr>
<td>General memory management and performance options</td>
<td>301</td>
</tr>
<tr>
<td>Configuring additional memory</td>
<td>301</td>
</tr>
<tr>
<td>Increasing logging information</td>
<td>301</td>
</tr>
<tr>
<td>Troubleshooting</td>
<td>303</td>
</tr>
<tr>
<td>Cannot load objects from an Oracle source database</td>
<td>303</td>
</tr>
<tr>
<td>Warning message</td>
<td>303</td>
</tr>
<tr>
<td>Reference</td>
<td>304</td>
</tr>
<tr>
<td>Release notes</td>
<td>305</td>
</tr>
<tr>
<td>Release notes – 662</td>
<td>305</td>
</tr>
<tr>
<td>Release notes – 661</td>
<td>307</td>
</tr>
<tr>
<td>Release notes – 660</td>
<td>310</td>
</tr>
<tr>
<td>Release notes – 659</td>
<td>312</td>
</tr>
<tr>
<td>Release notes – 658</td>
<td>315</td>
</tr>
<tr>
<td>Release notes – 657</td>
<td>317</td>
</tr>
<tr>
<td>Release notes – 656</td>
<td>320</td>
</tr>
</tbody>
</table>
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, MariaDB, Oracle, SQL Server, PostgreSQL DB, an Amazon Aurora DB cluster, 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), for connections to an Amazon S3 bucket or another AWS resource. AWS SCT is also compliant with Federal Risk and Authorization Management Program (FedRAMP). For details about AWS and compliance efforts, see AWS services in scope by compliance program.

AWS SCT supports the following OLTP conversions.

<table>
<thead>
<tr>
<th>Source database</th>
<th>Target database</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Db2 for z/OS (version 12)</td>
<td>Amazon Aurora MySQL-Compatible Edition (Aurora MySQL), Amazon Aurora PostgreSQL-Compatible Edition (Aurora PostgreSQL), MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using IBM Db2 for z/OS as a source (p. 48).</td>
</tr>
<tr>
<td>IBM Db2 LUW (versions 9.1, 9.5, 9.7, 10.5, 11.1, and 11.5)</td>
<td>Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using IBM Db2 LUW as a source (p. 50).</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Azure SQL Database as a source (p. 46).</td>
</tr>
<tr>
<td>Microsoft SQL Server (version 2008 R2 and later)</td>
<td>Aurora MySQL, Aurora PostgreSQL, Babelfish for Aurora PostgreSQL (only for assessment reports), MariaDB, Microsoft SQL Server, MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using SQL Server as a source (p. 71).</td>
</tr>
<tr>
<td>MySQL (version 5.5 and later)</td>
<td>Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using MySQL as a source (p. 55).</td>
</tr>
</tbody>
</table>
You can migrate schema and data from MySQL to an Aurora MySQL DB cluster without using AWS SCT. For more information, see Migrating data to an Amazon Aurora DB cluster.

Oracle (version 10.2 and later)

Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, Oracle, PostgreSQL

For more information, see Using Oracle Database as a source (p. 57).

PostgreSQL (version 9.1 and later)

Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL

For more information, see Using PostgreSQL as a source (p. 69).

SAP ASE (versions 12.5, 15.0, 15.5, 15.7, and 16.0)

Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL

For more information, see Using SAP ASE (Sybase ASE) as a source (p. 98).

AWS SCT supports the following data warehouse conversions.

<table>
<thead>
<tr>
<th>Source data warehouse</th>
<th>Target data warehouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Redshift</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Amazon Redshift as a source for AWS SCT (p. 101).</td>
</tr>
<tr>
<td>Azure Synapse Analytics (version 10)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Azure Synapse Analytics as a source (p. 103).</td>
</tr>
<tr>
<td>Greenplum Database (version 4.3 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Greenplum Database as a source (p. 104).</td>
</tr>
<tr>
<td>Microsoft SQL Server (version 2008 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using SQL Server Data Warehouse as a source (p. 115).</td>
</tr>
<tr>
<td>Netezza (version 7.0.3 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Netezza as a source (p. 106).</td>
</tr>
<tr>
<td>Oracle (version 10.2 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Oracle Data Warehouse as a source (p. 109).</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 AWS SCT (p. 5).

For an introduction to the AWS SCT user interface, see Using the AWS SCT user interface (p. 14).

For information on the conversion process, see Converting database schemas using AWS SCT (p. 142).

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. 215).
- 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 AWS SCT with AWS DMS (p. 213).
- 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 AWS SCT extension packs (p. 296).
- 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 AWS SCT (p. 179).

### Table: Source data warehouse to Target data warehouse

<table>
<thead>
<tr>
<th>Source data warehouse</th>
<th>Target data warehouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata (version 13 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Teradata as a source (p. 117).</td>
</tr>
<tr>
<td>Vertica (version 7.2 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Vertica as a source (p. 119).</td>
</tr>
<tr>
<td>Snowflake (version 3)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Snowflake as a source (p. 111).</td>
</tr>
</tbody>
</table>

### Table: Source database to Target database

<table>
<thead>
<tr>
<th>Source database</th>
<th>Target database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Cassandra (versions 2.1.0, 2.1.20, and 3.11.4)</td>
<td>Amazon DynamoDB</td>
</tr>
<tr>
<td></td>
<td>For more information, see Using Apache Cassandra as a source (p. 45).</td>
</tr>
</tbody>
</table>
- 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 AWS SCT (p. 268).
- You can use AWS SCT to migrate extraction, transformation, and load (ETL) processes. For more information, see Converting extract, transform, and load (ETL) processes with AWS Schema Conversion Tool (p. 181).

### Providing feedback

You can provide feedback about AWS SCT. You can file a bug report, submit a feature request, or 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 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, Microsoft Windows, and Ubuntu Linux version 15.04. AWS SCT is supported only on 64-bit operating systems.

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 AWS SCT (p. 5)
- Verifying the AWS SCT file download (p. 6)
- Installing the required database drivers (p. 8)
- Updating AWS SCT (p. 13)

Installing AWS SCT

To install 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
   - 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>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.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Install instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fedora Linux</td>
<td>Run the following command in the folder that you downloaded the file to:</td>
</tr>
</tbody>
</table>
Installing previous versions of AWS SCT

You can download and install previous versions of AWS SCT starting from 1.0.625. To download a previous version, provide the version and OS information using the following format.

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

For example, to download AWS SCT version 661, do the following:

- Windows – https://d211wdu1froga6.cloudfront.net/builds/1.0/661/Windows/aws-schema-conversion-tool-1.0.zip
- Ubuntu – https://d211wdu1froga6.cloudfront.net/builds/1.0/661/Ubuntu/aws-schema-conversion-tool-1.0.zip
- Fedora – https://d211wdu1froga6.cloudfront.net/builds/1.0/661/Fedora/aws-schema-conversion-tool-1.0.zip

AWS supports the three latest versions of the AWS SCT application. We recommend that you use the latest version of AWS SCT.

Verifying the AWS SCT file download

There are several ways you can verify the distribution file of 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 following to verify the distribution file, based on the operating system where you installed the file.

This section includes the following topics.

Topics
- Verifying the checksum of the AWS SCT file (p. 7)
- Verifying the AWS SCT RPM files on Fedora (p. 7)
- Verifying the AWS SCT DEB files on Ubuntu (p. 8)
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:

Fedora   b4f5f66f91bfc1b312e2827e960691c269a9002cd1371cf1841593f88cbb5e6
Ubuntu   4315eb6644949dcd95932351f00399adb6c6cf64b9f30adda2e0ec903c54eca4
Windows  6e29679a3c53c5396a06d8d50f308981e4ec34bd0acdd608874470700a0ae9a23

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 Linux 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. Verify 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:

    rpm -q --qf "%{NAME}-%{VERSION}-%{RELEASE} \n %{SUMMARY} \n" gpg-pubkey-ea22abf6-5a21d30c

7. Check the RPM signature by running the following command:

    rpm --checksig -v aws-schema-conversion-tool-1.0.build number-1.x86_64.rpm
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:
   
   ```
gpg --import aws-dms-team@amazon.com.public.gpg-key
   ```
7. Verify the signature by running the following command:
   
   ```
gpg --verify aws-schema-conversion-tool-1.0.latest.deb.asc aws-schema-conversion-tool-1.0.build number.deb
   ```

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.

Installing the required database drivers

For AWS SCT to work correctly, install the JDBC drivers for your source and target database engines. If you use a virtual target database platform, you don't need to install the JDBC driver for your target database engine. For more information, see Using virtual targets (p. 123).

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. 11).

You can download the database drivers from the following locations.

**Important**

Install the latest version of the driver available. The following table includes the lowest version of database driver supported by AWS SCT.
## Installing the required database drivers

<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>Edition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon Aurora PostgreSQL-</td>
<td>postgresql-42.2.19.jar</td>
<td><a href="https://jdbc.postgresql.org/download/postgresql-42.2.19.jar">https://jdbc.postgresql.org/download/postgresql-42.2.19.jar</a></td>
</tr>
<tr>
<td>Compatible Edition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon Redshift</td>
<td>redshift-jdbc42-2.0.0.1.jar</td>
<td><a href="https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.0.0.1/redshift-jdbc42-2.0.0.1.zip">https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.0.0.1/redshift-jdbc42-2.0.0.1.zip</a></td>
</tr>
<tr>
<td>Greenplum Database</td>
<td>postgresql-42.2.19.jar</td>
<td><a href="https://jdbc.postgresql.org/download/postgresql-42.2.19.jar">https://jdbc.postgresql.org/download/postgresql-42.2.19.jar</a></td>
</tr>
<tr>
<td>IBM Db2 for z/OS</td>
<td>db2jcc-db2jcc4.jar</td>
<td><a href="https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads-db2-zos">https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads-db2-zos</a></td>
</tr>
<tr>
<td>IBM Db2 LUW</td>
<td>db2jcc-db2jcc4.jar</td>
<td><a href="https://www.ibm.com/support/pages/node/382667">https://www.ibm.com/support/pages/node/382667</a></td>
</tr>
<tr>
<td>Maria DB</td>
<td>mariadb-java-client-2.4.1.jar</td>
<td><a href="https://downloads.mariadb.com/Connectors/java/connector-java-2.4.1/mariadb-java-client-2.4.1.jar">https://downloads.mariadb.com/Connectors/java/connector-java-2.4.1/mariadb-java-client-2.4.1.jar</a></td>
</tr>
<tr>
<td>MySQL</td>
<td>mysql-connector-java-8.0.15.jar</td>
<td><a href="https://dev.mysql.com/downloads/connector/j/">https://dev.mysql.com/downloads/connector/j/</a></td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>postgresql-42.2.19.jar</td>
<td><a href="https://jdbc.postgresql.org/download/postgresql-42.2.19.jar">https://jdbc.postgresql.org/download/postgresql-42.2.19.jar</a></td>
</tr>
</tbody>
</table>

- Please use the client tools software.
- Install driver version 7.2.1, which is backwards compatible with data warehouse version 7.2.0.

- Driver versions 8 and later are supported.
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.

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

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

<table>
<thead>
<tr>
<th>Database engine</th>
<th>Installation commands</th>
</tr>
</thead>
</table>
   | 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 |

---

**Database engine**

- **SAP ASE (Sybase ASE)**
  - **Drivers**
    - jconn4.jar
  - **Download location**

- **Teradata**
  - **Drivers**
    - terajdbc4.jar
    - tdgssconfig.jar
  - **Download location**
    - [https://downloads.teradata.com/download/connectivity/jdbc-driver](https://downloads.teradata.com/download/connectivity/jdbc-driver)

- **Vertica**
  - **Drivers**
    - vertica-jdbc-9.1.1-0.jar
  - **Download location**

- **Snowflake**
  - **Drivers**
    - snowflake-jdbc-3.9.2.jar
  - **Download location**

---

**Version 1.0**

10
### 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 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.
## Storing driver paths in the global settings

### Global settings

<table>
<thead>
<tr>
<th>Category</th>
<th>Driver settings</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Drivers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance and memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JVM options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assessment Report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS service profiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Notifications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tree view</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Oracle driver path
- Microsoft SQL Server driver path
- SQL Server Windows Authentication library
- MySQL driver path
- PostgreSQL driver path
- Teradata drivers path
- Amazon Redshift driver path
- Netezza driver path
- Greenplum driver path
- Vertica driver path
- DB2 LUW driver path
- MariaDB driver path
- SAP ASE (Sybase ASE) driver path
- Snowflake driver path
3. When you are finished adding the driver paths, choose OK.

Updating AWS SCT

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

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

**To check for updates to AWS SCT**

1. When in 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 SCT user interface

Use the following topics to help you work with the AWS SCT user interface. For information on installing AWS SCT, see Installing, verifying, and updating AWS SCT (p. 5).

Topics

- The AWS SCT project window (p. 14)
- Starting AWS SCT (p. 15)
- Creating an AWS SCT project (p. 16)
- Saving and opening an AWS SCT project (p. 17)
- Adding database servers to an AWS SCT project (p. 17)
- Running AWS SCT in an offline mode (p. 18)
- Using AWS SCT tree filters (p. 18)
- Hiding schemas in the AWS SCT tree view (p. 21)
- Creating and reviewing the database migration assessment report (p. 23)
- Converting your schema (p. 30)
- Applying the converted schema to your target DB instance (p. 34)
- Storing AWS service profiles in the AWS SCT (p. 34)
- Using AWS Secrets Manager (p. 39)
- Storing database passwords (p. 39)
- Using the UNION ALL view for projects with partitioned tables (p. 39)
- Keyboard shortcuts for AWS SCT (p. 40)

The AWS SCT project window

The illustration following is what you see in 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 are displayed. These describe the source schema element and the SQL command to create that element in the source database.

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

Starting AWS SCT

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: /opt/aws-schema-conversion-tool/bin/AWSSchemaConversionTool</td>
</tr>
</tbody>
</table>
Creating an AWS SCT project

Use the following procedure to create an AWS Schema Conversion Tool project.

**To create your project**

1. Start the AWS Schema Conversion Tool.
2. On the **File** menu, choose **New project**. The **New project** dialog box appears.

3. Enter a name for your project, which is stored locally on your computer.
4. Enter the location for your local project file.
5. Choose **OK** to create your AWS SCT project.
6. Choose **Add source** to add a new source database to your AWS SCT project. You can add multiple source databases to your AWS SCT project.
7. Choose **Add target** to add a new target platform in your AWS SCT project. You can add multiple target platforms to your AWS SCT project.
8. Choose the source database schema in the left panel.
9. In the right panel, specify the target database platform for the selected source schema.
10. Choose **Create mapping**. This button becomes active after you choose the source database schema and the target database platform.

Now, your AWS SCT project is set up. You can save your project, create database migration assessment report, and convert your source database schemas.
Saving and opening an AWS SCT project

Use the following procedure to save an AWS Schema Conversion Tool project.

**To save your project**

1. Start the AWS Schema Conversion Tool.
2. On the **File** menu, choose **Save project**.

   AWS SCT saves the project in the folder, which you specified when you created the project.

Use the following procedure to open an existing AWS Schema Conversion Tool project.

**To open your project**

1. On the **File** menu, choose **Open project**. The **Open** dialog box appears.
2. Choose the project folder and then choose the Windows Script Component (*.sct) file.

If you open a project saved in AWS SCT version 1.0.655 or before, AWS SCT automatically creates mapping rules for all source database schemas to the target database platform. To add other target database platforms, delete existing mapping rules and then create new mapping rules. For more information on creating mapping rules, see *Creating mapping rules* (p. 122).

Adding database servers to an AWS SCT project

You can add multiple source and target database servers to an AWS Schema Conversion Tool project.

**To add a server to your project**

1. Start the AWS Schema Conversion Tool.
2. Create a new project or open an existing project.
3. Choose **Add source** from the menu to add a new source database.
4. Choose a database platform and specify database connection credentials. For more information on connecting to a source database, see *Sources for AWS SCT* (p. 43).

Use the following procedure to connect to your database.

**To connect to your database**

1. Open the context (right-click) menu for a database server, and then choose **Establish connection**.
   You can also choose **Connect to the server** at the top of your database schema tree.
2. Enter the password to connect to your source database server.
3. Choose **Test connection** to verify that AWS SCT can connect to your source database.
4. Choose **Connect** to connect to your source database.

Use the following procedure to remove a database server from your AWS SCT project.

**To remove a database server**

1. Choose the database server to remove.
2. Open the context (right-click) menu, and then choose Remove from project.

AWS SCT removes the selected database server, all mapping rules, conversion results, and other metadata related to this server.

Running AWS SCT in an offline mode

You can run AWS Schema Conversion Tool in an offline mode. Following, you can learn how to work with an existing AWS SCT project when disconnected from your source database.

AWS SCT doesn't require a connection to your source database to run the following operations:

- Add mapping rules.
- Create database migration assessment reports.
- Convert database schemas and code.
- Edit your source and converted code.
- Save your source and converted code as SQL scripts in a text file.

Before you use AWS SCT in an offline mode, connect to your source database, load metadata, and save your project. Open this project or disconnect from the source database server to use AWS SCT in an offline mode.

To run AWS SCT in an offline mode

1. Start the AWS Schema Conversion Tool and create a new project. For more information, see Creating an AWS SCT project (p. 16).
2. Add a source database server and connect to your source database. For more information, see Adding database servers to an AWS SCT project (p. 17).
3. Add a target database server or use a virtual target database platform. For more information, see Using virtual targets (p. 123).
4. Create a mapping rule to define the target database platform for your source database. For more information, see Creating mapping rules in AWS SCT (p. 122).
5. Choose View, and then choose Main view.
6. In the left panel that displays the objects of your source database, choose your source database schemas. Open the context (right-click) menu for the object, and then choose Load schema. This operation loads all source schema metadata into your AWS SCT project.

The Create report and Convert schema operations also load all source schema metadata into your AWS SCT project. If you ran one of these operations from the context menu, skip the Load schema operation.

7. On the File menu, choose Save project to save the source database metadata in your project.
8. Choose Disconnect from the server to disconnect from your source database. Now you can use AWS SCT in the offline mode.

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 changes only 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 that you want to apply the tree filter to.
3. Choose the filter icon.

The undo filter icon is grayed out because no filter is currently applied.
4. Enter the following information in the Filter dialog box. Options in the dialog box are different for each database engine.

<table>
<thead>
<tr>
<th>AWS SCT filter option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level</strong></td>
<td>Choose Categories to filter objects by categories. Choose Statuses to filter objects by statuses.</td>
</tr>
</tbody>
</table>
| **Type**              | For Categories in Level, choose the categories of filtered objects. Choose Any loaded to display objects from all categories. For Statuses in Level, choose the status of filtered objects. You can choose one of the following options:  
- Converted to display all converted objects  
- Has actions to display all objects that have conversion issues  
- Encrypted to display all encrypted objects |
| **Condition**         | For Categories in Level, choose the filtering condition between Like and Not like. For Statuses in Level, the filtering condition option isn't available. |
| **Value**             | For Categories in Level, enter the Value to filter the tree by this value. Use the percent (%) as a wildcard to display all objects. For Statuses in Level, choose the Value between True and False. |
| **And/Or**            | Choose AND or OR logical operators to apply multiple filter clauses. |
5. Choose **Add new clause** to add an additional filter clause. AWS SCT can apply multiple filter clauses using **AND** or **OR** logical operators.

6. 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.

7. 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 comma separated value (CSV) file or a JSON file that contains names or values that you want the tree filter to use. Open an existing AWS SCT project, connect to the database to apply the tree filter to, and then choose the filter icon.

To download an example of the file, choose **Download template**. Enter the file name and choose **Save**.

To download your existing filter settings, choose **Export**. Enter the file name and choose **Save**.

To import a file list for the tree filter, choose **Import**. Choose a file to import, and then choose **Open**. Choose **Apply**, and then choose **Close**.

**CSV files** have the following format:

- **object_type** is the type of object that you want to find.
- **database_name** is the name of database where this object exists.
- **schema_name** is the name of schema where this object exists.
- **object_name** is the object name.
- **import_type** specifies to include or exclude this item from the filter.

Use **JSON files** to describe complex filtering cases, such as nested rules. **JSON files** have the following format:

- **filterGroupType** is the type of filter rule (AND or OR logical operators) that applies to multiple filter clauses.
- **filterCategory** is the level of the filter (Categories or Statuses).
- **names** is the list of object names that applies for the Categories filter.
- **filterCondition** is the filtering condition (LIKE or NOT LIKE) that applies for the Categories filter.
- **transformName** is the status name that applies for the Status filter.
- **value** is the value to filter the tree by.
- **transformValue** is the value of the filter (TRUE or FALSE) that applies for the Status filter.

**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 **Vendor**, choose database platform.
   - Choose **Hide empty schemas** to hide empty schemas for the selected database platform.
   - Choose **Hide empty databases** to hide empty databases for the selected database platform.
Managing the database migration assessment report

- For **Hide system databases/schemas**, choose system databases and schemas by name to hide them.
- For **Hide user-defined databases/schemas**, enter the names of user-defined databases and schemas that you want to hide, and then choose **Add**. The names are case insensitive.

5. Choose **OK**.

Creating and reviewing the database migration assessment report

The **database migration assessment report** summarizes all of the action items for schemas 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 a database migration assessment report after you add the source databases and target platforms to your project and specify mapping rules.

**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. Choose one of the action items in 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 to remove.

An example of an assessment report summary is shown following.
Database migration assessment report

Source database: GOLD_TEST_SS_PG

Microsoft SQL Server 2019 (RTM-CU10) (KB5001090) - 15.0.4123.1 (X64) Mar 22 2021 18:10:24

Executive summary

We completed the analysis of your Microsoft SQL Server source database and estimate that 90% of the database storage objects and 77% of the database code objects can be converted to Amazon RDS for PostgreSQL. Based on the source code syntax analysis, we estimate 94% (based on # lines of code) of your code can be converted to Amazon RDS for PostgreSQL.

Migration guidance for database objects that could not be converted automatically can be found here.

Database objects with conversion actions for Amazon RDS for PostgreSQL

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identified 529 (90%) database storage objects and 1,475 (96%) database code objects that can be converted.

We found 7 encrypted object(s).

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

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

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple action items will have a higher complexity.

Figure: Conversion statistics for database storage objects
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 CSV file. When you choose this option, AWS SCT creates three CSV files. These files contain the following information:

- A list of conversion action items with recommended actions.
- A summary of conversion action items with an estimate of the effort required to convert an occurrence of the action item.
- An executive summary with a number of action items categorized by the estimated time to convert.
Database objects with conversion actions for Amazon RDS PostgreSQL

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

We found 7 encrypted object(s).

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

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

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

Figure: Conversion statistics for database storage objects
Converting your schema

After you added source and target databases to your project and created mapping rules, you can convert your source database schemas. Use the following procedure to convert schema.

**To convert your schema**

1. Choose **View**, and then choose **Main view**.

2. In the left panel that displays the schema from your source database, choose schemas to convert. Open the context (right-click) menu for the object, and then choose **Convert schema**.
Converting your 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 database instance. The planned schema is part of your project. If you choose a converted schema item, you can see the planned schema command in the panel at lower center for your target database 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 database instance when you choose to apply your converted schema.
AWS Schema Conversion Tool User Guide
Converting your schema

create procedure POSITION_UPDATE_CA
    @InputPosNo nvarchar(100) = '',
    @posFlags bigint = 0,
    @posFlagsMask bigint = 0
AS
update p
set    p.Flags = p.Flags |
from Position p
inner join

CREATE OR REPLACE PROCEDURE test_db
AS
$BODY$
BEGIN
    PERFORM test_db
    INSERT INTO "parent" (parent_id) SELECT *
    FROM UNNEST
    UPDATE test_db
    SET flags = p.flags
    FROM "par_Input" WHERE p.parent_id = current
    return_code := 0
    RETURN;
$BODY$
Applying the converted schema to your target DB instance

You can apply the converted database schema to your target 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**
The following procedure overwrites the existing target schema. Be careful not to overwrite schemas unintentionally. Be careful not to overwrite schemas in your target DB instance that you have already modified, or you overwrite those changes.

**To apply the converted database schema to your target database 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.

Storing AWS service profiles in the AWS SCT

You can store your AWS credentials in 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 (Amazon RDS), and AWS Database Migration Service (AWS DMS).

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.
Storing AWS credentials

AWS service profiles

Define one or more service profiles to enable SCT to access AWS. To specify the default profile for the current project in the Global settings:

+ Add a new AWS service profile

Profile name: profile
AWS access key: aws_access_key
AWS secret key: ********
Region: US East (N. Virginia)
AWS S3 bucket folder:
Use FIPS endpoint for S3

Default profile: profile
3. Choose **Add a 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>Profile name</strong></td>
<td>Enter a name for your profile.</td>
</tr>
<tr>
<td><strong>AWS access key</strong></td>
<td>Enter your AWS access key.</td>
</tr>
<tr>
<td><strong>AWS secret key</strong></td>
<td>Enter your AWS secret key.</td>
</tr>
<tr>
<td><strong>Region</strong></td>
<td>Choose the AWS Region for your profile.</td>
</tr>
<tr>
<td><strong>AWS S3 bucket folder</strong></td>
<td>Choose the Amazon S3 bucket for your profile. You need to specify a bucket only if you are using a feature that connects to Amazon S3.</td>
</tr>
</tbody>
</table>

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.
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 and create a new project.
3. Choose the Project environment tab.
4. Choose Add a new AWS service profile to add a new profile. Then for AWS service profile, choose the profile that you want to associate with the project.
5. Choose OK to close the Project settings dialog box. You can also choose Cancel to cancel your changes.
Using AWS Secrets Manager

AWS SCT can use database credentials that you store in AWS Secrets Manager. You can fill in all values in the database connection dialog box from Secrets Manager. To use Secrets Manager, make sure that you store AWS profiles in the AWS Schema Conversion Tool.

For more information about using AWS Secrets Manager, see What is AWS Secrets Manager? in the AWS Secrets Manager User Guide. For more information about storing AWS profiles, see Storing AWS service profiles in the AWS SCT (p. 34).

To retrieve database credentials from Secrets Manager

1. Start the AWS Schema Conversion Tool and create a new project.
2. Choose Add source or Add target to add a new database to your project.
3. Choose a database platform and then choose Next.
4. For AWS Secret, choose the secret you want to use.
5. Choose Populate. Then AWS SCT fills in all values in the database connection dialog box.
6. Choose Test connection to verify that AWS SCT can connect to your database.
7. Choose Connect to connect to your database.

AWS SCT supports secrets that have the following structure.

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

In this structure, the username and password values are required, and all other values are optional. Make sure that the values that you store in Secrets Manager include all database credentials.

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 the UNION ALL view for projects with partitioned tables

If a source table is partitioned, AWS SCT creates n target tables, where n is the number of partitions on the source table. AWS SCT creates a UNION ALL view on top of the target tables to represent the source table. If you use an AWS SCT data extractor to migrate your data, the source table partitions will be extracted and loaded in parallel by separate subtasks.
To use Union All view for a project

1. Start AWS SCT. Create a new project or open an existing AWS SCT project.
2. On the Settings menu, choose Conversion settings.
3. Choose a pair of OLAP databases from the list at the top.
4. Turn on Use Union all view?

5. Choose OK to save the settings and close the Conversion settings dialog box.

Keyboard shortcuts for AWS SCT

The following are the keyboard shortcuts that you can use with 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>Keyboard shortcut</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Ctrl+W</td>
<td>Create a new project by using the wizard.</td>
</tr>
<tr>
<td>Ctrl+M</td>
<td>Create a new multiserver assessment.</td>
</tr>
<tr>
<td>Ctrl+L</td>
<td>Add a new source database.</td>
</tr>
<tr>
<td>Ctrl+R</td>
<td>Add a new target database.</td>
</tr>
<tr>
<td>Ctrl+F4</td>
<td>Close an open project.</td>
</tr>
<tr>
<td>F1</td>
<td>Open the <em>AWS SCT User Guide</em>.</td>
</tr>
</tbody>
</table>
Getting started with AWS SCT

You can use the AWS Schema Conversion Tool (AWS SCT) to convert the schema for a source database located either on-premises or hosted by Amazon Web Services (AWS). You can convert your source schema to a schema for any supported database that is hosted by 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 AWS SCT (p. 5).
2. Install an AWS SCT agent, if needed. AWS SCT agents are only required for certain migration scenarios, such as between heterogeneous sources and targets. For more information, see Using data extraction agents (p. 215).
3. Familiarize yourself with the user interface of AWS SCT. For more information, see Using the AWS SCT user interface (p. 14).
4. Create an AWS SCT project. Connect to your source and target databases. For more information about connecting to your source database, see Sources for AWS SCT (p. 43).
5. Create mapping rules. For more information about mapping rules, see Creating mapping rules in AWS SCT (p. 122).
6. 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. 23).
7. 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 AWS SCT (p. 142).

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 using AWS SCT (p. 157).
8. Applying the schema conversion to your target. For more information about applying a source schema conversion, see Using the AWS SCT user interface (p. 14).
9. You can also use AWS SCT to convert SQL stored procedures and other application code. For more information, see Converting application SQL using AWS SCT (p. 268)

You can also use AWS SCT to migrate your data from a source database to an Amazon-managed database. For examples, see Using data extraction agents (p. 215).
Sources for AWS SCT

AWS Schema Conversion Tool (AWS SCT) can convert schemas from the following source databases and data warehouses to a target database or data warehouse. For information about permissions, connections, and what AWS SCT can convert for use with the target database or data warehouse, see details in the topics listed following.

Encryption information

Encrypting Amazon RDS connections (p. 43)

Database sources

- Using Apache Cassandra as a source (p. 45)
- Using Azure SQL Database as a source (p. 46)
- Using IBM Db2 LUW as a source (p. 50)
- Using IBM Db2 for z/OS as a source (p. 48)
- Using MySQL as a source (p. 55)
- Using Oracle Database as a source (p. 57)
- Using PostgreSQL as a source (p. 69)
- Using SAP ASE (Sybase ASE) as a source (p. 98)
- Using SQL Server as a source (p. 71)

Data warehouse sources

- Using Amazon Redshift as a source (p. 101)
- Using Azure Synapse Analytics as a source (p. 103)
- Using Greenplum Database as a source (p. 104)
- Using Netezza as a source (p. 106)
- Using Oracle Data Warehouse as a source (p. 109)
- Using SQL Server Data Warehouse as a source (p. 115)
- Using Snowflake as a source (p. 111)
- Using Teradata as a source (p. 117)
- Using Vertica as a source (p. 119)

Encrypting Amazon RDS and Amazon Aurora connections in AWS SCT

To open encrypted connections to Amazon RDS or Amazon Aurora databases from an application, you need to import AWS root certificates into some form of key storage. You can download the root certificates from AWS at Using SSL/TLS to Encrypt a Connection to a DB Instance in the Amazon RDS User Guide.

Two options are available, a root certificate that works for all AWS Regions and a certificate bundle that contains both the old and new root certificates.

Depending on which you want to use, follow the steps in one of the two following procedures.
To import the certificate or certificates into the Windows system storage

1. Download a certificate or certificates from one of the following sources:

2. In your Windows search window, enter Manage computer certificates. When prompted as to whether to let the application make changes to your computer, choose Yes.

3. When the certificates window opens, if needed expand Certificated - Local Computer so you can see the list of certificates. Open the context (right-click) menu for Trusted Root Certification Authorities, then choose All Tasks, Import.

4. Choose Next, then Browse, and find the *.pem file that you downloaded in step 1. Choose Open to select the certificate file, choose Next, and then choose Finish.
   
   **Note**
   To find the file, change the file type in the browse window to All files (*.*), because .pem is not a standard certificate extension.

5. In the Microsoft Management Console, expand Certificates. Then expand Trusted Root Certification Authorities, choose Certificates, and find the certificate to confirm that it exists.

6. Restart your computer.

To import the certificate or certificates into the Java KeyStore

1. Download the certificate or certificates from one of the following sources:

2. If you downloaded the certificate bundle, split it into individual certificates files. To do so, place each certificate block, beginning with -----BEGIN CERTIFICATE----- and ending with -----END CERTIFICATE----- into a separate *.pem files. After you have created a separate *.pem file for each certificate, you can safely remove the certificate bundle file.

3. Open a command window or terminal session in the directory where you downloaded the certificate, and run the following command for every *.pem file that you created in the previous step.

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

**Example**

The following example assumes that you downloaded the rds-ca-2019-root.pem file.

```
Using Apache Cassandra as a source for AWS SCT

You can use AWS SCT to convert keyspaces from Apache Cassandra to Amazon DynamoDB.

Connecting to Apache Cassandra as a source

Use the following procedure to connect to your Apache Cassandra source database with the AWS Schema Conversion Tool.

To connect to an Apache Cassandra source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Cassandra, then choose Next.
The Add source dialog box appears.

3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.

4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
     2. Choose Populate to automatically fill in all values in the database connection dialog box from Secrets Manager.
     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).
   - To enter the Apache Cassandra source database connection information manually, 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>Enter the Domain Name Service (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Enter the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. 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>Choose 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 store: The trust store to use.</td>
</tr>
<tr>
<td></td>
<td>• Key store: The key store to use.</td>
</tr>
<tr>
<td>Store password</td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.</td>
</tr>
</tbody>
</table>

5. Choose Test Connection to verify that AWS SCT can connect to your source database.

6. Choose Connect to connect to your source database.

Using Azure SQL Database as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from Azure SQL Database to the following targets:
Privileges for Azure SQL Database as a source

The privileges required for Azure SQL Database as a source are listed following:

- VIEW DEFINITION
- VIEW DATABASE STATE

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

Connecting to Azure SQL Database as a source

Use the following procedure to connect to your Azure SQL Database source database with the AWS Schema Conversion Tool.

To connect to an Azure SQL Database source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Azure SQL Database, then choose Next.
   
   The Add source dialog box appears.
3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
     2. Choose Populate to automatically fill in all values in the database connection dialog box from Secrets Manager.

     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).

   - To enter the Azure SQL Database source database connection information manually, 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>Enter the Domain Name Service (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the database name to connect to.</td>
</tr>
</tbody>
</table>

Version 1.0
For this parameter | Do this
--- | ---
User name and Password | Enter the user name and password to connect to your source database. AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.

Store password | AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.

Using IBM Db2 for z/OS as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from IBM Db2 for z/OS to the following targets.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

Privileges for Db2 for z/OS as a source

The privileges needed to connect to a Db2 for z/OS database and read system catalogs and tables are listed following:

- SELECT ON SYSIBM.LOCATIONS
- SELECT ON SYSIBM.SYSCHECKS
- SELECT ON SYSIBM.SYSCOLUMNS
- SELECT ON SYSIBM.SYSDATABASE
- SELECT ON SYSIBM.SYSDATATYPES
- SELECT ON SYSIBM.SYSINDEXES
- SELECT ON SYSIBM.SYSKEYCOLUSE
- SELECT ON SYSIBM.SYSFOREIGNKEYS
- SELECT ON SYSIBM.SYSINDEXES
- SELECT ON SYSIBM.SYSKEYCOLUSE
- SELECT ON SYSIBM.SYSKEYS
- SELECT ON SYSIBM.SYSKEYTARGETS
- SELECT ON SYSIBM.SYSAUTOCREATETABLES
- SELECT ON SYSIBM.SYSAUTOINDEXES
To convert Db2 for z/OS tables to PostgreSQL partitioned tables, gather statistics on table spaces and tables in your database using the `RUNSTATS` utility as shown following.

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

In the preceding example, replace the `YOURDB` placeholder with the name of the source database.

### Connecting to Db2 for z/OS as a source

Use the following procedure to connect to your Db2 for z/OS source database with AWS SCT.

**To connect to an IBM Db2 for z/OS source database**

1. In the AWS Schema Conversion Tool, choose **Add source**.
2. Choose **Db2 for z/OS**, then choose **Next**.

   The **Add source** dialog box appears.

3. For **Connection name**, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

        For information about using database credentials from Secrets Manager, see **Using AWS Secrets Manager** (p. 39).
   - To enter the IBM Db2 for z/OS source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Location</td>
<td>Enter the unique name of the Db2 location you want to access.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Enter the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. 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>Choose 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 store: The location of a trust store containing certificates. For this location to appear here, make sure to add it in Global settings.</td>
</tr>
<tr>
<td>Store password</td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td>Db2 for z/OS driver path</td>
<td>Enter 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. 11).</td>
</tr>
</tbody>
</table>

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.

### Using IBM Db2 LUW as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from IBM for Linux, Unix, and Windows (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, 11.1, and 11.5.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
• Amazon RDS for MariaDB

Privileges for Db2 LUW as a source

The privileges needed to connect to a Db2 LUW database, to check available privileges and read schema metadata for a source are listed following:

• Privilege needed to establish a connection:
  • CONNECT ON DATABASE

• Privilege needed to run SQL statements:
  • EXECUTE ON PACKAGE NULLID.SYSSH200

• Privileges needed to get instance-level information:
  • EXECUTE ON FUNCTION SYSPROC.ENV_GET_INST_INFO
  • SELECT ON SYSIBMADM.ENV_INST_INFO
  • SELECT ON SYSIBMADM.ENV_SYS_INFO

• Privileges needed to check privileges granted through roles, groups, and authorities:
  • EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID
  • EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID
  • EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID
  • SELECT ON SYSIBMADM.PRIVILEGES

• Privileges needed on system catalogs and tables:
  • SELECT ON SYSCAT.ATTRIBUTES
  • SELECT ON SYSCAT.CHECKS
  • SELECT ON SYSCAT.COLIDENTATTRIBUTES
  • SELECT ON SYSCAT.COLUMNS
  • SELECT ON SYSCAT.DATAPARTITIONEXPRESSION
  • SELECT ON SYSCAT.DATAPARTITIONS
  • SELECT ON SYSCAT.DATATYPEDEP
  • SELECT ON SYSCAT.DATATYPES
  • SELECT ON SYSCAT.HIERARCHIES
  • SELECT ON SYSCAT.INDEXCOLUSE
  • SELECT ON SYSCAT.INDEXES
  • SELECT ON SYSCAT.INDEXPARTITIONS
  • SELECT ON SYSCAT.KEYCOLUSE
  • SELECT ON SYSCAT.MODULEOBJECTS
  • SELECT ON SYSCAT.MODELES
  • SELECT ON SYSCAT.NICKNAMES
  • SELECT ON SYSCAT.PERIODS
  • SELECT ON SYSCAT.REFERENCES
  • SELECT ON SYSCAT.ROUTINEPARMS
  • SELECT ON SYSCAT.ROUTINES
  • SELECT ON SYSCAT.ROWFIELDS
  • SELECT ON SYSCAT.SCHEMATA
  • SELECT ON SYSCAT.SEQUENCES
  • SELECT ON SYSCAT.TABCONST
  • SELECT ON SYSCAT.TABLES
• SELECT ON SYSCAT.TRIGGERS
• SELECT ON SYSCAT.VARIABLEDEP
• SELECT ON SYSCAT.VARIABLES
• SELECT ON SYSCAT.VIEWS
• SELECT ON SYSIBM.SYSDUMMY1

• 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:
  • USAGE ON WORKLOAD SYSDEFAULTUSERWORKLOAD

To run queries, you need to create system temporary tablespaces 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 Db2 LUW as a source

Use the following procedure to connect to your Db2 LUW source database with the AWS Schema Conversion Tool.

To connect to a Db2 LUW source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Db2 LUW, then choose Next.

   The Add source dialog box appears.
3. For **Connection name**, enter a name for your database. AWS SCT displays this name in the tree in the left panel.

4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.
   
   For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).

   - To enter the IBM Db2 LUW source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the Db2 LUW database.</td>
</tr>
</tbody>
</table>
   | **User name and Password** | Enter the user name and password to connect to your source database server.  
   AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project.  
   To guard against exposing the password for your source database, AWS SCT doesn't store the password by default.  
   If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.  |
   | **Use SSL**            | Choose 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:  
   **Trust store**: The location of a trust store containing certificates. For this location to appear here, make sure to add it in **Global settings**. |
   | **Store password**     | AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password. |
   | **Db2 LUW driver path** | Enter 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. 11). |

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.

6. Choose **Connect** to connect to your source database.
Converting Db2 LUW to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition

When you migrate IBM Db2 LUW to PostgreSQL, 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.

- **INSTEAD OF triggers** – AWS SCT supports these.

Converting Db2 LUW partitioned tables to PostgreSQL version 10 partitioned tables

AWS SCT can convert Db2 LUW tables to partitioned tables in PostgreSQL 10. There are several restrictions when converting a Db2 LUW partitioned table to PostgreSQL:

- You can create a partitioned table with a nullable column in Db2 LUW, and you can specify a partition to store NULL values. However, PostgreSQL doesn’t support NULL values for RANGE partitioning.

- Db2 LUW can use an INCLUSIVE or EXCLUSIVE clause to set range boundary values. PostgreSQL only supports INCLUSIVE for a starting boundary and EXCLUSIVE for an ending boundary. The converted partition name is in the format `<original_table_name>_<original_partition_name>`.

- You can create primary or unique keys for partitioned tables in Db2 LUW. PostgreSQL requires you to create primary or unique key for each partition directly. Primary or unique key constraints must be removed from the parent table. The converted key name is in the format `<original_key_name>_<original_partition_name>`.

- You can create a foreign key constraint from and to a partitioned table in Db2 LUW. However, PostgreSQL doesn’t support foreign keys references in partitioned tables. PostgreSQL also doesn’t support foreign key references from a partitioned table to another table.

- You can create an index on a partitioned table in Db2 LUW. However, PostgreSQL requires you to create an index for each partition directly. Indexes must be removed from the parent table. The converted index name is in the format `<original_index_name>_<original_partition_name>`.

- You must define row triggers on individual partitions, not on the partitioned table. Triggers must be removed from the parent table. The converted trigger name is in the format `<original_trigger_name>_<original_partition_name>`.
Using MySQL as a source for AWS SCT

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

- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon RDS for MySQL

For more information, see the following sections:

Topics
- Privileges for MySQL as a source (p. 55)
- Connecting to MySQL as a source (p. 55)

Privileges for MySQL as a source

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

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

Connecting to MySQL as a source

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

To connect to a MySQL source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose MySQL, then choose Next.

   The Add source dialog box appears.

3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:

   - To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
     2. Choose Populate to automatically fill in all values in the database connection dialog box from Secrets Manager.

     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).

   - To enter the MySQL source database connection information manually, use the instructions in the following table.
For this parameter | Do this
--- | ---
Server name | Enter the Domain Name System (DNS) name or IP address of your source database server.
Server port | Enter the port used to connect to your source database server.
User name and Password | Enter the user name and password to connect to your source database server.
AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.
Use SSL | Choose this option to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:
* **Require SSL**: Choose this option 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 Configuring MySQL to Use 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.

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 | Enter 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. 11).

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.
Using Oracle Database as a source for AWS SCT

You can use AWS SCT to convert schemas, database code objects, and application code from Oracle Database to the following targets:

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon RDS for Oracle
- Amazon RDS for MariaDB

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 more information, see the following.

Topics
- Privileges for Oracle as a source (p. 57)
- Connecting to Oracle as a source (p. 57)
- Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL (p. 59)
- Converting Oracle to Amazon RDS for MySQL or Amazon Aurora MySQL (p. 62)
- Converting Oracle to Amazon RDS for Oracle (p. 66)

Privileges for Oracle as a source

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

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

Connecting to Oracle as a source

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

To connect to an Oracle source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Oracle, then choose Next.
   
   The Add source dialog box appears.
3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
• To use database credentials from Secrets Manager, use the following instructions:
  1. For **AWS Secret**, choose the name of the secret.
  2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

For information about using database credentials from Secrets Manager, see [Using AWS Secrets Manager](p. 39).

• To enter the Oracle source database connection information manually, use the instructions in the following table.

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</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 Domain Name System (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><code>SELECT sys_context('userenv','instance_name') AS SID FROM dual;</code></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>After you choose the TNS file, AWS SCT adds all Oracle database connections from the file to the <strong>TNS alias</strong> list.</td>
</tr>
<tr>
<td></td>
<td>Choose this option to connect to Oracle Real Application Clusters (RAC).</td>
</tr>
<tr>
<td></td>
<td>• <strong>TNS alias</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>
<tr>
<td></td>
<td>• <strong>TNS connect identifier</strong>: The identifier for the registered TNS connection information.</td>
</tr>
</tbody>
</table>
### For this parameter | Do this
--- | ---
User name and Password | Enter the user name and password to connect to your source database server.

The first time you connect to the Oracle database, you enter the path to the Oracle Driver file (ojdbc8.jar). You can download the file at [http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html](http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html). Make sure to 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**.

AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.

Use SSL | Choose this option to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the **SSL** tab:

- **SSL authentication**: Select this option to use SSL authentication by certificate instead of user name and password. Set up your trust store and key store in **Settings**, **Global settings**, **Security**.
- **Trust store**: The trust store to use.
- **Key store**: The key store to use.

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

Oracle driver path | Enter 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 in the connection dialog box. For more information, see **Storing driver paths in the global settings** (p. 11).

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.

## Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL

When you convert an Oracle to RDS for PostgreSQL or Amazon Aurora PostgreSQL, be aware of the following.
Topics

- Converting Oracle sequences (p. 60)
- Converting Oracle ROWID (p. 60)
- Converting Oracle dynamic SQL (p. 61)
- Converting Oracle partitions (p. 62)

When converting Oracle system objects to PostgreSQL, AWS SCT performs conversions as shown in the following table.

<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>

You can use AWS SCT to convert Oracle SQL*Plus files to psql, which is a terminal-based front-end to PostgreSQL. For more information, see Converting application SQL using AWS SCT (p. 268).

Converting Oracle sequences

AWS SCT converts sequences from Oracle to PostgreSQL. If you use sequences to maintain integrity constraints, make sure that new values of a migrated sequence don’t overlap the existing values.

**To populate converted sequences with the last value from the source database**

1. Open your AWS SCT project with Oracle as the source.
2. Choose Settings, and then choose Conversion settings.
3. From the upper list, choose Oracle, and then choose Oracle – PostgreSQL. AWS SCT displays all available settings for Oracle to PostgreSQL conversion.
4. Choose Populate converted sequences with the last value generated on the source side.
5. Choose OK to save the settings and close the Conversion settings dialog box.

Converting Oracle ROWID

In an Oracle database, the ROWID pseudocolumn contains the address of the table row. The ROWID pseudocolumn is unique to Oracle, so AWS SCT converts the ROWID pseudocolumn to a data column on PostgreSQL. By using this conversion, you can keep the ROWID information.

When converting the ROWID pseudocolumn, AWS SCT can create a data column with the bigint data type. If no primary key exists, AWS SCT sets the ROWID column as the primary key. If a primary key exists, AWS SCT sets the ROWID column with a unique constraint.

If your source database code includes operations with ROWID, which you can't run using a numeric data type, AWS SCT can create a data column with the character varying data type.

**To create a data column for Oracle ROWID for a project**

1. Open your AWS SCT project with Oracle as the source.
2. Choose Settings, and then choose Conversion settings.
3. From the upper list, choose Oracle, and then choose Oracle – PostgreSQL. AWS SCT displays all available settings for Oracle to PostgreSQL conversion.

4. For Generate row ID, do one of the following:
   - Choose Generate as identity to create a numeric data column.
   - Choose Generate as character domain type to create a character data column.

5. Choose OK to save the settings and close the Conversion settings dialog box.

Converting Oracle dynamic SQL

Oracle provides two ways to implement dynamic SQL: using an EXECUTE IMMEDIATE statement or calling procedures in the DBMS_SQL package. If your source Oracle database includes objects with dynamic SQL, use AWS SCT to convert Oracle dynamic SQL statements to PostgreSQL.

To convert Oracle dynamic SQL to PostgreSQL

1. Open your AWS SCT project with Oracle as the source.
2. Choose a database object that uses dynamic SQL in the Oracle source tree view.
3. Open the context (right-click) menu for the object, choose Convert schema, and agree to replace the objects if they exist. The following screenshot shows the converted procedure below the Oracle procedure with dynamic SQL.
Converting Oracle partitions

AWS SCT currently supports the following partitioning methods:

- Range
- List
- Multicolumn range
- Hash
- Composite (list-list, range-list, list-range, list-hash, range-hash, hash-hash)

Converting Oracle to Amazon RDS for MySQL or Amazon Aurora MySQL

When you convert Oracle to RDS for MySQL or Aurora MySQL, to change the order that statements run in, you can use a GOTO statement and a label. Any PL/SQL statements that follow a GOTO statement are skipped, and processing continues at the label. You can use GOTO statements and labels anywhere within a procedure, batch, or statement block. You can also nest GOTO statements.

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><code>BEGIN</code></td>
<td><code>BEGIN</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>label1:</code></td>
</tr>
<tr>
<td><code>statement1;</code></td>
<td><code>BEGIN</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>GOTO label1;</code></td>
<td><code>....</code></td>
</tr>
<tr>
<td><code>statement2;</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>LEAVE label1;</code></td>
</tr>
<tr>
<td><code>label1:</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>Statement3;</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>END;</code></td>
</tr>
<tr>
<td><code>END</code></td>
<td><code>Statement3;</code></td>
</tr>
<tr>
<td></td>
<td><code>END</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BEGIN</code></td>
<td><code>BEGIN</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>label1:</code></td>
</tr>
<tr>
<td><code>statement1;</code></td>
<td><code>BEGIN</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>label1:</code></td>
<td><code>....</code></td>
</tr>
<tr>
<td><code>statement2;</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>GLAVE label1;</code></td>
</tr>
<tr>
<td><code>GOTO label1;</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>statement3;</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>END LOOP;</code></td>
</tr>
<tr>
<td><code>statement4;</code></td>
<td><code>statement1;</code></td>
</tr>
<tr>
<td><code>....</code></td>
<td><code>END</code></td>
</tr>
</tbody>
</table>

Version 1.0
62
Converting the WITH statement in Oracle to Amazon RDS for MySQL or Amazon Aurora (MySQL)

You use the WITH clause (subquery_factoring) in Oracle to assign a name (query_name) to a subquery block. You can then reference the subquery block multiple places in the query by specifying query_name. If a subquery block doesn’t contain links or parameters (local, procedure, function, package), then AWS SCT converts the clause to a view or a temporary table.

The advantage of converting the clause to a temporary table is that repeated references to the subquery might be more efficient. The greater efficiency is because the data is easily retrieved from the temporary table rather than being required by each reference. You can emulate this by using additional views or a temporary table. The view name uses the format `<procedure_name>$<subselect_alias>`.

You can find examples in the table following.
<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL; END LOOP;</td>
<td>IF done THEN</td>
</tr>
<tr>
<td></td>
<td>LEAVE read_label;</td>
</tr>
<tr>
<td></td>
<td>END IF;</td>
</tr>
<tr>
<td></td>
<td>BEGIN</td>
</tr>
<tr>
<td></td>
<td>END;</td>
</tr>
<tr>
<td></td>
<td>END LOOP;</td>
</tr>
<tr>
<td></td>
<td>CLOSE cur;</td>
</tr>
<tr>
<td></td>
<td>END;</td>
</tr>
</tbody>
</table>
CREATE PROCEDURE TEST_ORA_PG.P_WITHSeleccione_REGULAR_MULT_01
AS
BEGIN
  FOR cur IN (WITH deptempl AS
    (SELECT id, name, surname, lastname, state, dept_id
    FROM test_ora_pg.dept_employees
    WHERE state = 1),
    dept AS
    (SELECT id deptid, parent_id, name deptname
    FROM test_ora_pg.department)
    SELECT deptempl.*, dept.*
    FROM deptempl, dept
    WHERE deptempl.dept_id = dept.deptid)
  LOOP
    NULL;
  END LOOP;
CREATE VIEW TEST_ORA_PG.`P_WITHSeleccione_REGULAR_MULT_01$deptempl`
(id, name, surname, lastname, state, dept_id)
AS
(SELECT id, name, surname, lastname, state, dept_id
 FROM test_ora_pg.dept_employees
 WHERE state = 1);
CREATE VIEW TEST_ORA_PG.`P_WITHSeleccione_REGULAR_MULT_01$dept`
(deptid, parent_id, deptname)
AS
(SELECT id deptid, parent_id, name deptname
 FROM test_ora_pg.department);
CREATE PROCEDURE test_ora_pg.P_WITHSeleccione_REGULAR_MULT_01()
BEGIN
  DECLARE var$ID DOUBLE;
  DECLARE var$NAME VARCHAR (30);
  DECLARE var$SURNAME VARCHAR (30);
  DECLARE var$LASTNAME VARCHAR (30);
  DECLARE var$STATE DOUBLE;
  DECLARE var$DEPT_ID DOUBLE;
  DECLARE var$deptid DOUBLE;
  DECLARE var$PARENT_ID DOUBLE;
  DECLARE var$deptname VARCHAR (200);
  DECLARE done INT DEFAULT FALSE;
  DECLARE cur CURSOR FOR SELECT
    deptempl.*, dept.*
    FROM TEST_ORA_PG.`P_WITHSeleccione_REGULAR_MULT_01$deptempl`
      AS deptempl,
    TEST_ORA_PG.`P_WITHSeleccione_REGULAR_MULT_01$dept`
      AS dept
    WHERE deptempl.DEPT_ID = dept.DEPTID;
  DECLARE CONTINUE HANDLER FOR NOT FOUND
    SET done := TRUE;
  OPEN cur;
  read_label:
  LOOP
    FETCH cur INTO var$ID, var$NAME, var$SURNAME,
      var$LASTNAME, var$STATE, var$DEPT_ID, var$deptid,
      var$PARENT_ID, var$deptname;
    IF done THEN
      LEAVE read_label;
    END IF;
    BEGIN
    END;
  END LOOP;
  CLOSE cur;
END;
Converting Oracle to Amazon RDS for Oracle

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

<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE PROCEDURE TEST_ORA_PG.P_WITH_SELECT_VAR_CROSS_02(p_state IN NUMBER) AS</td>
<td>CREATE VIEW TEST_ORA_PG.'P_WITH_SELECT_VAR_CROSS_01$emp' (id, name, surname, lastname, state, dept_id) AS</td>
</tr>
<tr>
<td>l_dept_id NUMBER := 10;</td>
<td>(SELECT id, name, surname, lastname, state, dept_id</td>
</tr>
<tr>
<td>BEGIN</td>
<td>FROM TEST_ORA_PG.DEPT_EMPLOYEES WHERE DEPT_ID &gt; 10);</td>
</tr>
<tr>
<td>FOR cur IN (</td>
<td>CREATE PROCEDURE test_ora_pg.P_WITH_SELECT_VAR_CROSS_02(IN par_P_STATE DOUBLE)</td>
</tr>
<tr>
<td>WITH emp AS</td>
<td>BEGIN</td>
</tr>
<tr>
<td>(SELECT id, name, surname, lastname, state, dept_id</td>
<td>DECLARE var_l_dept_id DOUBLE DEFAULT 10;</td>
</tr>
<tr>
<td>FROM test_ora_pg.dept_employees WHERE dept_id &gt; 10</td>
<td>DECLARE var$ID DOUBLE;</td>
</tr>
<tr>
<td>active_emp AS</td>
<td>DECLARE done INT DEFAULT FALSE;</td>
</tr>
<tr>
<td>FROM test_ora_pg.dept_employees</td>
<td>DECLARE cur CURSOR FOR SELECT * FROM (SELECT</td>
</tr>
<tr>
<td>WHERE dept_id &gt; 10</td>
<td>id, name, surname, lastname, state, dept_id) FROM TEST_ORA_PG.DEPT_EMPLOYEES</td>
</tr>
<tr>
<td>) LOOP</td>
<td>WHERE DEPT_ID &gt; 10);</td>
</tr>
<tr>
<td>END LOOP;</td>
<td>END;</td>
</tr>
<tr>
<td>END;</td>
<td>END;</td>
</tr>
</tbody>
</table>

- Version 1.0
• 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 a 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 Amazon RDS for Oracle user role, including advanced filtering and column privileges.

• AWS SCT supports converting Oracle jobs into jobs that can run on Amazon RDS for Oracle. There are a few limitations to the conversion, including the following:

  • Executable jobs are not supported.
  • Schedule jobs that use the ANYDATA data type as an argument are not supported.

• Oracle Real Application Clusters (RAC) One Node is an option to the Oracle Database Enterprise Edition that was introduced with Oracle Database 11g Release 2. Amazon RDS for Oracle doesn’t support the RAC feature. For high availability, use Amazon RDS Multi-AZ.

In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica. This functionality provides data redundancy, eliminates I/O freezes, and minimizes latency spikes during system backups.

• Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial data in an Oracle database. Oracle Locator provides capabilities that are typically required to support internet and wireless service-based applications and partner-based GIS solutions. Oracle Locator is a limited subset of Oracle Spatial.

To use Oracle Spatial and Oracle Locator features, add the SPATIAL option or LOCATOR option (mutually exclusive) to the option group of your DB instance.

There are some prerequisites to using Oracle Spatial and Oracle Locator on an Amazon RDS for Oracle DB instance:

• The instance should use Oracle Enterprise Edition version 12.1.0.2.v6 or later, or 11.2.0.4.v10 or later.

• The instance should be inside a virtual private cloud (VPC).

• The instance should be the DB instance class that can support the Oracle feature. For example, Oracle Spatial is not supported for the db.m1.small, db.t1.micro, db.t2.micro, or db.t2.small DB instance classes. For more information, see DB instance class support for Oracle.

• The instance must have the Auto Minor Version Upgrade option enabled. Amazon RDS updates your DB instance to the latest Oracle PSU if there are security vulnerabilities with a CVSS score of 9+ or other announced security vulnerabilities.
Settings for Oracle DB instances.

- If your DB instance is version 11.2.0.4.v10 or later, you must install the XMLDB option. For more information, see Oracle XML DB.
- You should have an Oracle Spatial license from Oracle. For more information, see Oracle Spatial and Graph in the Oracle documentation.
- Data Guard is included with Oracle Database Enterprise Edition. For high availability, use Amazon RDS Multi-AZ feature.

In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica. This functionality provides data redundancy, eliminates I/O freezes, and minimizes latency spikes during system backups.

- AWS SCT supports converting Oracle DBMS_SCHEDULER objects when migrating to Amazon RDS for Oracle. The AWS SCT assessment report indicates if a schedule object can be converted. For more information on using schedule objects with Amazon RDS, see the Amazon RDS documentation.
- 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 @dlink 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.

- The AWS SCT assessment report provides server metrics for the conversion. These metrics about your Oracle instance include the following:
  - Computation and memory capacity of the target DB instance.
  - Unsupported Oracle features such as Real Application Clusters that Amazon RDS doesn't support.
  - Disk read-write load
  - Average total disk throughput
  - Server information such as server name, OS, host name, and character set.

Limitations when converting Oracle to Amazon RDS for Oracle

Some limitations you should consider when migrating Oracle schema and code to Amazon RDS for Oracle:

- 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.

- Amazon RDS for Oracle supports traditional auditing, fine-grained auditing using the DBMS_FGA package, and Oracle Unified Auditing.
- Amazon RDS for Oracle doesn’t support change data capture (CDC). To do CDC during and after a database migration, use AWS Database Migration Service.

Using PostgreSQL as a source for AWS SCT

You can use AWS SCT to convert schemas, database code objects, and application code from PostgreSQL to the following targets:

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

For more information, see the following sections:

Topics
- Privileges for PostgreSQL as a source database (p. 69)
- Connecting to PostgreSQL as a source (p. 69)

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.

To connect to a PostgreSQL source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose PostgreSQL, then choose Next.
   The Add source dialog box appears.
3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

For information about using database credentials from Secrets Manager, see [Using AWS Secrets Manager](#) (p. 39).

- To enter the PostgreSQL source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the PostgreSQL database.</td>
</tr>
</tbody>
</table>
| **User name and Password** | Enter the user name and password to connect to your source database server. AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project.  
                           | To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed. |
| **Use SSL**              | Choose this option to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the **SSL** tab:  
                           | - **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. For this location to appear in the **Global settings** section, make sure to add it. |
| **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. |
| **PostgreSQL driver path** | Enter 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. 11). |

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.

6. Choose **Connect** to connect to your source database.
Using Microsoft SQL Server as a source for AWS SCT

You can use AWS SCT to convert schemas, database code objects, and application code from SQL Server to the following targets:

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon RDS for SQL Server
- Amazon RDS for MariaDB

You can use AWS SCT to create an assessment report for the migration of schemas, database code objects, and application code from SQL Server to Babelfish for Aurora PostgreSQL.

For more information, see the following sections:

Topics
- Privileges for Microsoft SQL Server as a source (p. 71)
- Using Windows Authentication when using Microsoft SQL Server as a source (p. 71)
- Connecting to SQL Server as a source (p. 73)
- Converting SQL Server to MySQL (p. 74)
- Converting SQL Server to PostgreSQL (p. 75)
- Converting SQL Server to Amazon RDS for SQL Server (p. 98)

Privileges for 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.

Mode enables both Windows Authentication and SQL Server Authentication. Windows Authentication is always available and cannot be disabled. For more information about Windows Authentication, see the Microsoft Windows documentation.

The possible example for creating a user in TEST_DB is shown following.

```
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, are not automatically specified 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`

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

- Declare two variables that 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 the folder where your JDK/JRE is running. You may copy to lib folder, bin folder, and so on. As an example, you might copy 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 have only the SQLJDBC4.jar file. Remove any other sqljdbc*.jar files from that folder (or copy them to another folder). If you are adding the driver as part of your program, ensure that you add only SQLJDBC4.jar as the driver to use.

- Copy `sqljdbc_auth.dll` the file in the folder with your application.
Connecting to SQL Server as a source

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

To connect to a Microsoft SQL Server source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Microsoft SQL Server, then choose Next.

The Add source dialog box appears.
3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
     2. Choose Populate to automatically fill in all values in the database connection dialog box from Secrets Manager.

   For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).
   - To enter the Microsoft SQL Server source database connection information manually, 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>Enter the Domain Name Service (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Instance name</td>
<td>Enter 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>Authentication</td>
<td>Choose the authentication type from Windows Authentication and SQL Server Authentication.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Enter the user name and password to connect to your source database server. AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are</td>
</tr>
</tbody>
</table>
For this parameter | Do this
--- | ---
prompted for the password to connect to your source database as needed.

**Use SSL**
Choose this option to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:
- **Trust server certificate**: Select this option to trust the server certificate.
- **Trust store**: The location of a trust store containing certificates. For this location to appear in the Global settings section, make sure to add it.

**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**
Enter 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. 11).

**Windows Authentication library**
Enter the path to the sqljdbc_auth.dll file. By default, this file is installed in the following location:

```
<installation directory of the JDBC driver>sqljdbc_<version>\<language>\auth
```

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.

## Converting SQL Server 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.
### SQL Server statement

<table>
<thead>
<tr>
<th>SQL Server statement</th>
<th>MySQL statement</th>
</tr>
</thead>
</table>
| BEGIN
... statement1;
... GOTO label1;
statement2;
... label1:
Statement3;
... END   | BEGIN
label1:
BEGIN
... statement1;
... LEAVE label1;
statement2;
... END;
Statement3;
... END   |

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

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

MySQL doesn't support multi-statement table-valued functions. AWS SCT simulates table-valued functions during a conversion by creating temporary tables and rewriting statements to use these temporary tables.

**Converting SQL Server to PostgreSQL**

You can use the SQL Server to PostgreSQL extension pack in AWS SCT. This extension pack emulates SQL Server database functions in the converted PostgreSQL code. Use the SQL Server to PostgreSQL...
extension pack to emulate SQL Server Agent and SQL Server Database Mail. For more information about extension packs, see Using AWS SCT extension packs (p. 296).

When you convert a Microsoft SQL Server database to Amazon Aurora PostgreSQL-Compatible Edition (Aurora PostgreSQL) or Amazon Relational Database Service for PostgreSQL (Amazon RDS for PostgreSQL), be aware of the following.

Topics
• Converting SQL Server partitions to PostgreSQL version 10 partitions (p. 76)
• Migration considerations (p. 76)
• Using an AWS SCT extension pack to emulate SQL Server Agent in PostgreSQL (p. 79)
• Using an AWS SCT extension pack to emulate SQL Server Database Mail in PostgreSQL (p. 88)

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.

Migration considerations

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 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.

**SQL Server GOTO statements and the converted PostgreSQL statements**

<table>
<thead>
<tr>
<th>SQL Server statement</th>
<th>PostgreSQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN .... statement1; .... GOTO label1; statement2; .... label1: Statement3; .... END</td>
<td>BEGIN label1: BEGIN .... statement1; .... EXIT label1; statement2; .... END; Statement3; .... END</td>
</tr>
<tr>
<td>BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END</td>
<td>BEGIN .... statement1; .... label1: LOOP statement2; .... CONTINUE label1; EXIT label1; END LOOP; statement3; .... statement4; .... END</td>
</tr>
<tr>
<td>BEGIN .... statement1; .... label1: statement2; .... statement3; .... statement4; .... END</td>
<td>BEGIN .... statement1; .... label1: BEGIN statement2; .... statement3; .... statement4; .... END; END</td>
</tr>
</tbody>
</table>

- PostgreSQL doesn’t support a MERGE statement. AWS SCT emulates the behavior of a MERGE statement in the following ways:
  - By INSERT ON CONFLICT construction.
  - By using the UPDATE FROM DML statement, such as MERGE without a WHEN NOT MATCHED clause.
• By using CURSOR, such as with a MERGE with DELETE clause or by using a complex MERGE ON condition statement.
• 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.
• When migrating from Microsoft SQL Server to PostgreSQL, the built-in SUSER_SNAME function is converted as follows:
  • SUSER_SNAME – Returns the login name associated with a security identification number (SID).
  • SUSER_SNAME(<server_user_sid>) – Not supported.
  • SUSER_SNAME() CURRENT_USER – Returns the user name of the current execution context.
  • SUSER_SNAME(NULL) – Returns NULL.
• Converting table-valued functions is supported. Table-valued functions return a table and can take the place of a table in a query.
• PATINDEX returns the starting position of the first occurrence of a pattern in a specified expression on all valid text and character data types. It returns zeros if the pattern is not found. 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>).
• In SQL Server, a user-defined table type is a type that represents the definition of a table structure. You use a user-defined table type to declare table-value parameters for stored procedures or functions. You can also use a user-defined table type to declare table variables that you want to use in a batch or in the body of a stored procedure or function. AWS SCT emulated this type in PostgreSQL by creating a temporary table.

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>
<tr>
<td>SYS.FOREIGN_KEY_COLUMNS</td>
<td>AWS_SQLSERVER_EXT.SYS_FOREIGN_KEY_COLUMNS</td>
</tr>
<tr>
<td>SYS.KEY_CONSTRAINTS</td>
<td>AWS_SQLSERVER_EXT.SYS_KEY_CONSTRAINTS</td>
</tr>
<tr>
<td>SYS.IDENTITY_COLUMNS</td>
<td>AWS_SQLSERVER_EXT.SYS.IDENTITY_COLUMNS</td>
</tr>
<tr>
<td>SYS.PROCEDURES</td>
<td>AWS_SQLSERVER_EXT.SYS_PROCEDURES</td>
</tr>
<tr>
<td>SYS.INDEXES</td>
<td>AWS_SQLSERVER_EXT.SYS_INDEXES</td>
</tr>
</tbody>
</table>

Version 1.0
78
Using an AWS SCT extension pack to emulate SQL Server Agent in PostgreSQL

SQL Server Agent is a Microsoft Windows service that runs SQL Server jobs. SQL Server Agent runs jobs on a schedule, in response to a specific event, or on demand. For more information about SQL Server Agent, see Microsoft technical documentation.

PostgreSQL doesn’t have an equivalent for SQL Server Agent. To emulate the SQL Server Agent features, AWS SCT creates an extension pack. This extension pack uses AWS Lambda and Amazon CloudWatch. AWS Lambda implements the interface that you use to manage schedules and run jobs. Amazon CloudWatch maintains the schedule rules.

AWS Lambda and Amazon CloudWatch use a JSON parameter to interact. This JSON parameter has the following structure.

```
{
  "mode": mode,
  "parameters": {
    "list of parameters"
  },
```

### MS SQL Server use cases | PostgreSQL substitution

<table>
<thead>
<tr>
<th>MS SQL Server use cases</th>
<th>PostgreSQL substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS.SYSINDEXES</td>
<td>AWS_SQLSERVER_EXT.SYS_SYSINDEXES</td>
</tr>
<tr>
<td>SYS.OBJECTS</td>
<td>AWS_SQLSERVER_EXT.SYS_OBJECTS</td>
</tr>
<tr>
<td>SYS.ALL_OBJECTS</td>
<td>AWS_SQLSERVER_EXT.SYS_ALL_OBJECTS</td>
</tr>
<tr>
<td>SYS.SYSOBJECTS</td>
<td>AWS_SQLSERVER_EXT.SYS_SYSOBJECTS</td>
</tr>
<tr>
<td>SYS.SQL_MODULES</td>
<td>AWS_SQLSERVER_EXT.SYS_SQL_MODULES</td>
</tr>
<tr>
<td>SYS.DATABASES</td>
<td>AWS_SQLSERVER_EXT.SYS_DATABASES</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.SCHEMATA</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_SCHEMATA</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.VIEWS</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_VIEWS</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.TABLES</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLES</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.COLUMNS</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_COLUMNS</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.CHECK_CONSTRAINTS</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CHECK_CONSTRAINTS</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMAREFERENTIAL_CONSTRAINTS</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.KEY_COLUMN_USAGE</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_KEY_COLUMN_USAGE</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_TABLE_USAGE</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_COLUMN_USAGE</td>
</tr>
<tr>
<td>INFORMATION_SCHEMA.ROUTINES</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_ROUTINES</td>
</tr>
<tr>
<td>SYS.SYSPROCESSES</td>
<td>AWS_SQLSERVER_EXT.SYS_SYSPROCESSES</td>
</tr>
<tr>
<td>sys.system_objects</td>
<td>AWS_SQLSERVER_EXT.SYS_SYSTEM_OBJECTS</td>
</tr>
</tbody>
</table>
"callback": \textit{procedure name}

In the preceding example, \textit{mode} is the type of the task and \textit{list of parameters} is a set of parameters that depend on the type of the task. Also, \textit{procedure name} is the name of the procedure that runs after the task is completed.

\textbf{AWS SCT} uses one Lambda function to control and run jobs. The CloudWatch rule starts the run of the job and provides the necessary information to start the job. When the CloudWatch rule triggers, it starts the Lambda function using the parameters from the rule.

To create a simple job that calls a procedure, use the following format.

\begin{verbatim}
{
    "mode": "run_job",
    "parameters": {
        "vendor": "mysql",
        "cmd": "lambda_db.nightly_job"
    }
}
\end{verbatim}

To create a job with several steps, use the following format.

\begin{verbatim}
{
    "mode": "run_job",
    "parameters": {
        "job_name": "Job1",
        "enabled": "true",
        "start_step_id": 1,
        "notify_level_email": [0|1|2|3],
        "notify_email": email,
        "delete_level": [0|1|2|3],
        "job_callback": "ProcCallBackJob(job_name, code, message)",
        "step_callback": "ProcCallBackStep(job_name, step_id, code, message)"
    },
    "steps": [
    {
        "id": 1,
        "cmd": "ProcStep1",
        "cmdexec_success_code": 0,
        "on_success_action": [2|3|4],
        "on_success_step_id": 1,
        "on_fail_action": 0,
        "on_fail_step_id": 0,
        "retry_attempts": number,
        "retry_interval": number
    },
    {
        "id": 2,
        "cmd": "ProcStep2",
        "cmdexec_success_code": 0,
        "on_success_action": [1|2|3|4],
        "on_success_step_id": 0,
        "on_fail_action": 0,
        "on_fail_step_id": 0,
        "retry_attempts": number,
        "retry_interval": number
    },
    ...]
}
\end{verbatim}
To emulate the SQL Server Agent behavior in PostgreSQL, the AWS SCT extension pack also creates the following tables and procedures.

### Tables that emulate SQL Server Agent in PostgreSQL

To emulate SQL Server Agent, the extension pack uses the following tables:

**sysjobs**
- Stores the information about the jobs.

**sysjobsteps**
- Stores the information about the steps of a job.

**sysschedules**
- Stores the information about the job schedules.

**sysjobschedules**
- Stores the schedule information for individual jobs.

**sysjobhistory**
- Stores the information about the runs of scheduled jobs.

### Procedures that emulate SQL Server Agent in PostgreSQL

To emulate SQL Server Agent, the extension pack uses the following procedures:

**sp_add_job**
- Adds a new job.

**sp_add_jobstep**
- Adds a step to a job.

**sp_add_schedule**
- Creates a new schedule rule in Amazon CloudWatch. You can use this schedule with any number of jobs.

**sp_attach_schedule**
- Sets a schedule for the selected job.

**sp_add_jobschedule**
- Creates a schedule rule for a job in Amazon CloudWatch and sets the target for this rule.

**sp_update_job**
- Updates the attributes of the previously created job.

**sp_update_jobstep**
- Updates the attributes of the step in a job.

**sp_update_schedule**
- Updates the attributes of a schedule rule in Amazon CloudWatch.

**sp_update_jobschedule**
- Updates the attributes of the schedule for the specified job.

**sp_delete_job**
- Deletes a job.
sp_delete_jobstep
Deletes a job step from a job.

sp_delete_schedule
Deletes a schedule.

sp_delete_jobschedule
Deletes the schedule rule for the specified job from Amazon CloudWatch.

sp_detach_schedule
Removes an association between a schedule and a job.

get_jobs, update_job
Internal procedures that interact with AWS Elastic Beanstalk.

sp_verify_job_date, sp_verify_job_time, sp_verify_job, sp_verify_jobstep, sp_verify_schedule, sp_verify_job_identifiers, sp_verify_schedule_identifiers
Internal procedures that check settings.

Syntax for procedures that emulate SQL Server Agent in PostgreSQL

The `aws_sqlserver_ext.sp_add_job` procedure in the extension pack emulates the `msdb.dbo.sp_add_job` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

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

The `aws_sqlserver_ext.sp_add_jobstep` procedure in the extension pack emulates the `msdb.dbo.sp_add_jobstep` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

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

The `aws_sqlserver_ext.sp_add_schedule` procedure in the extension pack emulates the `msdb.dbo.sp_add_schedule` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

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

The `aws_sqlserver_ext.sp_attach_schedule` procedure in the extension pack emulates the `msdb.dbo.sp_attach_schedule` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

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

The `aws_sqlserver_ext.sp_add_jobschedule` procedure in the extension pack emulates the `msdb.dbo.sp_add_jobschedule` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_name varchar = NULL::character varying,
par_enabled smallint = 1,
par_freq_type integer = 1,
par_freq_interval integer = 0,
par_freq_subday_type integer = 0,
par_freq_subday_interval integer = 0,
par_freq_relative_interval integer = 0,
par_freq_recurrence_factor integer = 0,
par_active_start_date integer = NULL::integer,
par_active_end_date integer = 99991231,
par_active_start_time integer = 0,
par_active_end_time integer = 235959,
inout par_schedule_id integer = NULL::integer,
par_automatic_post smallint = 1,
inout par_schedule_uid char = NULL::bpchar,
out returncode integer

The aws_sqlserver_ext.sp_delete_job procedure in the extension pack emulates the msdb.dbo.sp_delete_job procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_originating_server varchar = NULL::character varying,
par_delete_history smallint = 1,
par_delete_unused_schedule smallint = 1,
out returncode integer

The aws_sqlserver_ext.sp_delete_jobstep procedure in the extension pack emulates the msdb.dbo.sp_delete_jobstep procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_step_id integer = NULL::integer,
out returncode integer

The aws_sqlserver_ext.sp_delete_jobschedule procedure in the extension pack emulates the msdb.dbo.sp_delete_jobschedule procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_name varchar = NULL::character varying,
par_keep_schedule integer = 0,
par_automatic_post smallint = 1,
out returncode integer

The aws_sqlserver_ext.sp_delete_schedule procedure in the extension pack emulates the msdb.dbo.sp_delete_schedule procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_force_delete smallint = 0,
par_automatic_post smallint = 1,
out returncode integer

The aws_sqlserver_ext.sp_detach_schedule procedure in the extension pack emulates the msdb.dbo.sp_detach_schedule procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_delete_unused_schedule smallint = 0,
par_automatic_post smallint = 1,
out returncode integer

The `aws_sqlserver_ext.sp_update_job` procedure in the extension pack emulates the `msdb.dbo.sp_update_job` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

```sql
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_start_step_id integer = NULL::integer
par_category_name varchar = NULL::character varying
par_owner_login_name varchar = NULL::character varying
par_notify_level_eventlog integer = NULL::integer
par_notify_level_email integer = NULL::integer
par_notify_level_netsend integer = NULL::integer
par_notify_level_page integer = NULL::integer
par_delete_level integer = NULL::integer
par_automatic_post smallint = 1
out returncode integer
```

The `aws_sqlserver_ext.sp_update_jobschedule` procedure in the extension pack emulates the `msdb.dbo.sp_update_jobschedule` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

```sql
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_freq_type integer = NULL::integer
par_freq_interval integer = NULL::integer
par_freq_subday_type integer = NULL::integer
par_freq_subday_interval integer = NULL::integer
par_freq_relative_interval integer = NULL::integer
par_freq_recurrence_factor integer = NULL::integer
par_active_start_date integer = NULL::integer
par_active_end_date integer = NULL::integer
par_active_start_time integer = NULL::integer
par_active_end_time integer = NULL::integer
par_automatic_post smallint = 1
out returncode integer
```

The `aws_sqlserver_ext.sp_update_jobstep` procedure in the extension pack emulates the `msdb.dbo.sp_update_jobstep` procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

```sql
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_step_id integer = NULL::integer
par_step_name varchar = NULL::character varying
par_subsystem varchar = NULL::character varying
par_command text = NULL::text
par_additional_parameters text = NULL::text
par_cmdexec_success_code integer = NULL::integer
par_on_success_action smallint = NULL::smallint
par_on_success_step_id integer = NULL::integer
par_on_fail_action smallint = NULL::smallint
```
par_on_fail_step_id integer = NULL::integer
par_server varchar = NULL::character varying
par_database_name varchar = NULL::character varying
par_database_user_name varchar = NULL::character varying
par_retry_attempts integer = NULL::integer
par_retry_interval integer = NULL::integer
par_os_run_priority integer = NULL::integer
par_output_file_name varchar = NULL::character varying
par_flags integer = NULL::integer
par_proxy_id integer = NULL::integer
par_proxy_name varchar = NULL::character varying
out returncode integer

The aws_sqlserver_ext.sp_update_schedule procedure in the extension pack emulates the msdb.dbo.sp_update_schedule procedure. For more information about the source SQL Server Agent procedure, see Microsoft technical documentation.

par_schedule_id integer = NULL::integer
par_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_freq_type integer = NULL::integer
par_freq_interval integer = NULL::integer
par_freq_subday_type integer = NULL::integer
par_freq_subday_interval integer = NULL::integer
par_freq_relative_interval integer = NULL::integer
par_freq_recurrence_factor integer = NULL::integer
par_active_start_date integer = NULL::integer
par_active_end_date integer = NULL::integer
par_active_start_time integer = NULL::integer
par_active_end_time integer = NULL::integer
par_owner_login_name varchar = NULL::character varying
par_automatic_post smallint = 1
out returncode integer

Examples for using procedures that emulate SQL Server Agent in PostgreSQL

To add a new job, use the aws_sqlserver_ext.sp_add_job procedure as shown following.

```
SELECT * FROM aws_sqlserver_ext.sp_add_job (  
   par_job_name := 'test_job',
   par_enabled := 1::smallint,
   par_start_step_id := 1::integer,
   par_category_name := '[Uncategorized (Local)]',
   par_owner_login_name := 'sa');
```

To add a new job step, use the aws_sqlserver_ext.sp_add_jobstep procedure as shown following.

```
SELECT * FROM aws_sqlserver_ext.sp_add_jobstep (  
   par_job_name := 'test_job',
   par_step_id := 1::smallint,
   par_step_name := 'test_job_step1',
   par_subsystem := 'TSQL',
   par_command := 'EXECUTE [dbo].[PROC_TEST_JOB_STEP1];',
   par_server := NULL,
   par_database_name := 'GOLD_TEST_SS');
```

To add a simple schedule, use the aws_sqlserver_ext.sp_add_schedule procedure as shown following.

```
SELECT * FROM aws_sqlserver_ext.sp_add_schedule(
```
par_schedule_name := 'RunOnce',
par_freq_type := 1,
par_active_start_time := 233000);

To set a schedule for a job, use the `aws_sqlserver_ext.sp_attach_schedule` procedure as shown following.

```sql
SELECT * FROM aws_sqlserver_ext.sp_attach_schedule (
    par_job_name := 'test_job',
    par_schedule_name := 'NightlyJobs');
```

To create a schedule for a job, use the `aws_sqlserver_ext.sp_add_jobschedule` procedure as shown following.

```sql
SELECT * FROM aws_sqlserver_ext.sp_add_jobschedule (
    par_job_name := 'test_job2',
    par_name := 'test_schedule2',
    par_enabled := 1::smallint,
    par_freq_type := 4,
    par_freq_interval := 1,
    par_freq_subday_type := 4,
    par_freq_subday_interval := 1,
    par_freq_relative_interval := 0,
    par_freq_recurrence_factor := 0,
    par_active_start_date := 20100801,
    par_active_end_date := 99991231,
    par_active_start_time := 0,
    par_active_end_time := 0);
```

Use case examples for emulating SQL Server Agent in PostgreSQL

If your source database code uses SQL Server Agent to run jobs, you can use the SQL Server to PostgreSQL extension pack for AWS SCT to convert this code to PostgreSQL. The extension pack uses AWS Lambda functions to emulate the behavior of SQL Server Agent.

You can create a new AWS Lambda function or register an existing function.

**To create a new AWS Lambda function**

1. In AWS SCT, in the target database tree, open the context (right-click) menu, choose **Apply extension pack for**, and then choose **PostgreSQL**.
   
   The extension pack wizard appears.

2. On the **SQL Server Agent emulation service** tab, do the following:
   
   - Choose **Create an AWS Lambda function**.
   - For **Database login**, enter the name of the target database user.
   - For **Database password**, enter the password for the user name that you entered on the preceding step.
   - For **Python library folder**, enter the path to your Python library folder.
   - Choose **Create AWS Lambda function**, and then choose **Next**.

**To register an AWS Lambda function that you deployed earlier**

- Run the following script on your target database.

```sql
SELECT
```
In the preceding example, ARN is the Amazon Resource Name (ARN) of the deployed AWS Lambda function.

The following example creates a simple task that consists of one step. Every five minutes, this task runs the previously created job_example function. This function inserts records into the job_example_table table.

**To create this simple task**

1. Create a job using the `aws_sqlserver_ext.sp_add_job` function as shown following.

```sql
SELECT
  FROM aws_sqlserver_ext.sp_add_job(
    par_job_name := 'test_simple_job');
```

2. Create a job step using the `aws_sqlserver_ext.sp_add_jobstep` function as shown following.

```sql
SELECT
  FROM aws_sqlserver_ext.sp_add_jobstep(
    par_job_name := 'test_simple_job',
    par_step_name := 'test_simple_job_step1',
    par_command := 'PERFORM job_simple_example;');
```

The job step specifies what the function does.

3. Create a scheduler for the job using the `aws_sqlserver_ext.sp_add_jobschedule` function as shown following.

```sql
SELECT
  FROM aws_sqlserver_ext.sp_add_jobschedule(
    par_job_name := 'test_simple_job',
    par_name := 'test_schedule',
    par_freq_type := 4, /* Daily */
    par_freq_interval := 1, /* frequency_interval is unused */
    par_freq_subday_type := 4, /* Minutes */
    par_freq_subday_interval := 5 /* 5 minutes */);
```

The job step specifies what the function does.

To delete this job, use the `aws_sqlserver_ext.sp_delete_job` function as shown following.

```sql
PERFORM aws_sqlserver_ext.sp_delete_job(
  par_job_name := 'PeriodicJob1'::character varying,
  par_delete_history := 1::smallint,
  par_delete_unused_schedule := 1::smallint);
```

**Using an AWS SCT extension pack to emulate SQL Server Database Mail in PostgreSQL**

You can use SQL Server Database Mail to send email messages to users from the SQL Server Database Engine or Azure SQL Managed Instance. These email messages can contain query results or include files.
from any resource on your network. For more information about SQL Server Database Mail, see Microsoft technical documentation.

PostgreSQL doesn't have an equivalent for SQL Server Database Mail. To emulate the SQL Server Database Mail features, AWS SCT creates an extension pack. This extension pack uses AWS Lambda and Amazon Simple Email Service (Amazon SES). AWS Lambda provides users with an interface to interact with Amazon SES email sending service. To set up this interaction, add the Amazon Resource Name (ARN) of your Lambda function.

For a new email account, use the following command.

```sql
do
## begin
PERFORM sysmail_add_account_sp (  
  par_account_name := 'your_account_name',  
  par_email_address := 'your_account_email',  
  par_display_name := 'your_account_display_name',  
  par_mailserver_type := 'AWSLAMBDA'  
  par_mailserver_name := 'ARN'  
);  
end;  
## language plpgsql;
```

To add the ARN of your Lambda function to the existing email account, use the following command.

```sql
do
## begin
PERFORM sysmail_update_account_sp (  
  par_account_name := 'existing_account_name',  
  par_mailserver_type := 'AWSLAMBDA'  
  par_mailserver_name := 'ARN'  
);  
end;  
## language plpgsql;
```

In the preceding examples, `ARN` is the ARN of your Lambda function.

To emulate the SQL Server Database Mail behavior in PostgreSQL, the AWS SCT extension pack uses the following tables, views, and procedures.

## Tables that emulate SQL Server Database Mail in PostgreSQL

To emulate SQL Server Database Mail, the extension pack uses the following tables:

- **sysmail_account**
  Stores the information about the email accounts.

- **sysmail_profile**
  Stores the information about the user profiles.

- **sysmail_server**
  Stores the information about the email servers.

- **sysmail_mailitems**
  Stores the list of the email messages.
sysmail_attachments

Contains one row for each email attachment.

sysmail_log

Stores the service information about sending email messages.

sysmail_profileaccount

Stores the information about the user profiles and email accounts.

**Views that emulate SQL Server Database Mail in PostgreSQL**

To emulate SQL Server Database Mail, AWS SCT creates the following views in the PostgreSQL database to ensure compatibility. The extension pack doesn't use them, but your converted code can query these views.

sysmail_allitems

Includes a list of all emails.

sysmail_faileditems

Includes a list of emails that couldn't be sent.

sysmail_sentitems

Includes a list of sent emails.

sysmail_unsentitems

Includes a list of emails that aren't sent yet.

sysmail_mailattachments

Includes a list of attached files.

**Procedures that emulate SQL Server Database Mail in PostgreSQL**

To emulate SQL Server Database Mail, the extension pack uses the following procedures:

sp_send_dbmail

Sends an email to the specified recipients.

sysmail_add_profile_sp

Creates a new user profile.

sysmail_add_account_sp

Creates a new email account that stores such information as Simple Mail Transfer Protocol (SMTP) credentials, and so on.

sysmail_add_profileaccount_sp

Adds an email account to the specified user profile.

sysmail_update_profile_sp

Changes the attributes of the user profile such as description, name, and so on.

sysmail_update_account_sp

Changes the information in the existing email account.
sysmail_update_profileaccount_sp

Updates the email account information in the specified user profile.

sysmail_delete_profileaccount_sp

Removes an email account from the specified user profile.

sysmail_delete_account_sp

Deletes the email account.

sysmail_delete_profile_sp

Deletes the user profile.

sysmail_delete_mailitems_sp

Deletes emails from internal tables.

sysmail_help_profile_sp

Displays information about the user profile.

sysmail_help_account_sp

Displays information about the email account.

sysmail_help_profileaccount_sp

Displays information about email accounts associated with the user profile.

sysmail_dbmail_json

An internal procedure that generates JSON requests for AWS Lambda functions.

sysmail_verify_profile_sp, sysmail_verify_account_sp, sysmail_verify_addressparams_sp

Internal procedures that check settings.

sp_get_dbmail, sp_set_dbmail, sysmail_dbmail_xml

Deprecated internal procedures.

Syntax for procedures that emulate SQL Server Database Mail in PostgreSQL

The aws_sqlserver_ext.sp_send_dbmail procedure in the extension pack emulates the msdb.dbo.sp_send_dbmail procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_profile_name varchar = NULL::character varying,
par_recipients text = NULL::text,
par_copy_recipients text = NULL::text,
par_blind_copy_recipients text = NULL::text,
par_subject varchar = NULL::character varying,
par_body text = NULL::text,
par_body_format varchar = NULL::character varying,
par_importance varchar = 'NORMAL'::character varying,
par_sensitivity varchar = 'NORMAL'::character varying,
par_file_attachments text = NULL::text,
par_query text = NULL::text,
par_execute_query_database varchar = NULL::character varying,
par_attach_query_result_as_file smallint = 0,
par_query_attachment_filename varchar = NULL::character varying,
par_query_result_header smallint = 1,
par_query_result_width integer = 256,
par_query_result_separator VARCHAR = ' '::character varying,
```

Version 1.0

91
The `aws_sqlserver_ext.sysmail_delete_mailitems_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_delete_mailitems_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```sql
par_sent_before timestamp = NULL::timestamp without time zone,
par_sent_status varchar = NULL::character varying,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_add_profile_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_add_profile_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```sql
par_profile_name varchar,
par_description varchar = NULL::character varying,
out par_profile_id integer,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_add_account_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_add_account_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```sql
par_account_name varchar
par_email_address varchar
par_display_name varchar = NULL::character varying
par_replyto_address varchar = NULL::character varying
par_description varchar = NULL::character varying
par_mailserver_name varchar = NULL::character varying
par_mailserver_type varchar = 'SMTP'::bpchar
par_port integer = 25
par_username varchar = NULL::character varying
par_password varchar = NULL::character varying
par_use_default_credentials smallint = 0
par_enable_ssl smallint = 0
out par_account_id integer
out returncode integer
```

The `aws_sqlserver_ext.sysmail_add_profileaccount_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_add_profileaccount_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```sql
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_sequence_number integer = NULL::integer,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_help_profile_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_help_profile_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.
The `aws_sqlserver_ext.sysmail_update_profile_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_update_profile_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_delete_profile_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_delete_profile_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_description varchar = NULL::character varying,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_help_account_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_help_account_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_update_account_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_update_account_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_email_address varchar = NULL::character varying,
par_display_name varchar = NULL::character varying,
par_replyto_address varchar = NULL::character varying,
par_description varchar = NULL::character varying,
par_mailserver_name varchar = NULL::character varying,
par_mailserver_type varchar = NULL::character varying,
par_port integer = NULL::integer,
par_username varchar = NULL::character varying,
par_password varchar = NULL::character varying,
par_use_default_credentials smallint = NULL::smallint,
par_enable_ssl smallint = NULL::smallint,
par_timeout integer = NULL::integer,
par_no_credential_change smallint = NULL::smallint,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_delete_account_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_delete_account_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
```
out returncode integer

The `aws_sqlserver_ext.sysmail_help_profileaccount_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_help_profileaccount_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_update_profileaccount_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_update_profileaccount_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_sequence_number integer = NULL::integer,
out returncode integer
```

The `aws_sqlserver_ext.sysmail_delete_profileaccount_sp` procedure in the extension pack emulates the `msdb.dbo.sysmail_delete_profileaccount_sp` procedure. For more information about the source SQL Server Database Mail procedure, see Microsoft technical documentation.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

Examples for using procedures that emulate SQL Server Database Mail in PostgreSQL

To send an email, use the `aws_sqlserver_ext.sp_send_dbmail` procedure as shown following.

```
PERFORM sp_send_dbmail (  
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Automated Success Message',
    par_body := 'The stored procedure finished'
);  
```

The following example shows how to send an email with query results.

```
PERFORM sp_send_dbmail (  
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Account with id = 1',
    par_query := 'SELECT COUNT(*)FROM Account WHERE id = 1'
);  
```

The following example shows how to send an email with HTML code.

```
DECLARE var_tableHTML TEXT;
```
SET var_tableHTML := CONCAT('  
  <H1>Work Order Report</H1>
  ',  
  '<table border="1">
  ',  
  '<tr><th>Work Order ID</th><th>Product ID</th>',  
  '<th>Name</th><th>Order Qty</th><th>Due Date</th>',  
  '<th>Expected Revenue</th></tr>',  
  '</table>
  ');  
PERFORM sp_send_dbmail (  
  par_recipients := 'hello@rusgl.info',  
  par_subject := 'Work Order List',  
  par_body := var_tableHTML,  
  par_body_format := 'HTML'  
);  

To delete emails, use the `aws_sqlserver_ext.sysmail_delete_mailitems_sp` procedure as shown following.

DECLARE var_GETDATE datetime;  
SET var_GETDATE = NOW();  
PERFORM sysmail_delete_mailitems_sp (  
  par_sent_before := var_GETDATE  
);  

The following example shows how to delete the oldest emails.

PERFORM systemail_delete_mailitems_sp (  
  par_sent_before := '31.12.2015'  
);  

The following example shows how to delete all emails that can't be sent.

PERFORM sysmail_delete_mailitems_sp (  
  par_sent_status := 'failed'  
);  

To create a new user profile, use the `aws_sqlserver_ext.sysmail_add_profile_sp` procedure as shown following.

PERFORM sysmail_add_profile_sp (  
  profile_name := 'Administrator',  
  par_description := 'administrative mail'  
);  

The following example shows how to create a new profile and save the unique profile identifier in a variable.

DECLARE var_profileId INT;  
SELECT par_profile_id  
  FROM sysmail_add_profile_sp (  
    profile_name := 'Administrator',  
    par_description := 'Profile used for administrative mail.'  
  )  
  INTO var_profileId;  
SELECT var_profileId;  

To create a new email account, use the `aws_sqlserver_ext.sysmail_add_account_sp` procedure as shown following.
To add an email account to the user profile, use the `sysmail_add_account_sp` procedure as shown following.

```sql
PERFORM sysmail_add_account_sp (  
    par_account_name :='Audit Account',  
    par_email_address := 'dba@rusgl.info',  
    par_display_name := 'Test Automated Mailer',  
    par_description := 'Account for administrative e-mail.',  
    par_mailserver_type := 'AWSLAMBDA'  
);  
```

To add an email account to the user profile, use the `aws_sqlserver_ext.sysmail_add_profileaccount_sp` procedure as shown following.

```sql
PERFORM sysmail_add_profileaccount_sp (  
    par_account_name := 'Administrator',  
    par_account_name := 'Audit Account',  
    par_sequence_number := 1  
);  
```

**Use case examples for emulating SQL Server Database Mail in PostgreSQL**

If your source database code uses SQL Server Database Mail to send emails, you can use the AWS SCT extension pack to convert this code to PostgreSQL.

**To send an email from your PostgreSQL database**

1. Create and configure your AWS Lambda function.
2. Apply the AWS SCT extension pack.
3. Create a user profile using the `sysmail_add_profile_sp` function as shown following.
4. Create an email account using the `sysmail_add_account_sp` function as shown following.
5. Add this email account to your user profile using the `sysmail_add_profileaccount_sp` function as shown following.

```sql
CREATE OR REPLACE FUNCTION aws_sqlserver_ext.proc_dbmail_settings_msdb()  
RETURNS void  
AS  
$BODY$  
BEGIN  
PERFORM aws_sqlserver_ext.sysmail_add_profile_sp(  
    par_profile_name := 'Administrator',  
    par_description := 'administrative mail'  
);  
PERFORM aws_sqlserver_ext.sysmail_add_account_sp(  
    par_account_name := 'Audit Account',  
    par_description := 'Account for administrative e-mail.',  
    par_email_address := 'dba@rusgl.info',  
    par_display_name := 'Test Automated Mailer',  
    par_mailserver_type := 'AWSLAMBDA'  
    par_mailserver_name := 'your_ARN'  
);  
PERFORM aws_sqlserver_ext.sysmail_add_profileaccount_sp(  
    par_profile_name := 'Administrator',  
    par_account_name := 'Audit Account',  
    par_sequence_number := 1  
);  
END;  
$BODY$  
LANGUAGE plpgsql;  
```
6. Send an email using the `sp_send_dbmail` function as shown following.

```
CREATE OR REPLACE FUNCTION aws_sqlserver_ext.
    proc_dbmail_send_msdb()
RETURNS void
AS
    $BODY$
    BEGIN
        PERFORM aws_sqlserver_ext.sp_send_dbmail(
            par_profile_name := 'Administrator',
            par_recipients := 'hello@rusgl.info',
            par_body := 'The stored procedure finished',
            par_subject := 'Automated Success Message'
        );
    END;
    $BODY$
LANGUAGE plpgsql;
```

To view the information about all user profiles, use the `sysmail_help_profile_sp` procedure as shown following.

```
SELECT FROM aws_sqlserver_ext.sysmail_help_profile_sp();
```

The following example displays the information about the specific user profile.

```
select from aws_sqlserver_ext.sysmail_help_profile_sp(par_profile_id := 1);
select from aws_sqlserver_ext.sysmail_help_profile_sp(par_profile_name := 'Administrator');
```

To view the information about all email accounts, use the `sysmail_help_account_sp` procedure as shown following.

```
select from aws_sqlserver_ext.sysmail_help_account_sp();
```

The following example displays the information about the specific email account.

```
select from aws_sqlserver_ext.sysmail_help_account_sp(par_account_id := 1);
select from aws_sqlserver_ext.sysmail_help_account_sp(par_account_name := 'Audit Account');
```

To view the information about all email accounts that are associated with the user profiles, use the `sysmail_help_profileaccount_sp` procedure as shown following.

```
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp();
```

The following example filters the records by identifier, profile name, or account name.

```
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_id := 1);
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_id := 1,
    par_account_id := 1);
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_name :=
    'Administrator');
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_account_name := 'Audit Account');
```

To change the user profile name or description, use the `sysmail_update_profile_sp` procedure as shown following.
To change the email account settings, use the `sysmail_update_account_sp` procedure as shown following.

```sql
select aws_sqlserver_ext.sysmail_update_account_sp(
    par_account_name := 'Audit Account',
    par_mailserver_type := 'AWSLAMBDA'
);
```

**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 with SQL Server Reporting Services (SSRS), SQL Server Analysis Services (SSAS), and SQL Server Integration Services (SSIS).
- Amazon RDS currently doesn't support SQL Server Service Broker or additional T-SQL endpoints that require you to run the CREATE ENDPOINT command.
- Amazon RDS has limited support for linked servers. When converting SQL Server application code that uses linked servers, AWS SCT converts the application code. However, make sure to review the behavior of objects that use linked servers before you run the converted code.
- Always on is used.
- The AWS SCT assessment report provides server metrics for the conversion. These metrics about your SQL Server instance include the following:
  - Data mirroring is used.
  - SQL Server Log Shipping is configured.
  - Failover cluster is used.
  - Database Mail is configured.
  - Full Text Search Service is used. Amazon RDS for SQL Server has a limited full text search, and it does not support semantic search.
  - Data Quality Service (DQS) is installed. Amazon RDS doesn't support DQS so we recommend that you install SQL Server on an Amazon EC2 instance.

**Using SAP ASE (Sybase ASE) as a source for AWS SCT**

You can use AWS SCT to convert schemas, database code objects, and application code from SAP (Sybase) Adaptive Server Enterprise (ASE) to the following targets:

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for MariaDB
Privileges for SAP ASE as a source database

To use an SAP ASE database as a source, you create a database user and grant permissions. To do this, take the following steps.

Create and configure a database user

1. Connect to the source database.
2. Create a database user with the following commands. Provide a password for the new user.

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

3. For every database you are going to migrate, grant the following privileges.

```
USE <database_name>
sp_adduser min_privs
grant select on dbo.sysusers to min_privs
grant select on dbo.sysobjects to min_privs
grant select on dbo.sysindexes to min_privs
grant select on dbo.syscolumns to min_privs
grant select on dbo.sysreferences to min_privs
grant select on dbo.sysindexes to min_privs
grant select on dbo.syscolumns to min_privs
grant select on dbo.sysreferences to min_privs
grant select on dbo.sysobjects to min_privs
grant select on dbo.sysusers to min_privs
grant select on dbo.syspartitionkeys to min_privs
grant select on dbo.sysconstraints to min_privs
grant select on dbo.systypes to min_privs
grant select on dbo.sysqueryplans to min_privs
```

Connecting to SAP ASE (Sybase) as a source

Use the following procedure to connect to your SAP ASE source database with the AWS Schema Conversion Tool.

To connect to a SAP ASE source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose SAP ASE, then choose Next.

   The Add source dialog box appears.

3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:

- To use database credentials from Secrets Manager, use the following instructions:
  1. For **AWS Secret**, choose the name of the secret.
  2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

  For information about using database credentials from Secrets Manager, see **Using AWS Secrets Manager (p. 39)**.

- To enter the SAP ASE source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the SAP ASE database.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Enter 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 choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. 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>Choose this option 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>1. <strong>Verify server certificate</strong>: Select this option to verify the server certificate by using a trust store.</td>
</tr>
<tr>
<td></td>
<td>2. <strong>Trust store</strong>: The location of a trust store containing certificates.</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>SAP ASE driver path</strong></td>
<td>Enter 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>
</tbody>
</table>

  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. 11)**.

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.

6. Choose **Connect** to connect to your source database.
Data warehouse sources for AWS Schema Conversion Tool

AWS SCT can convert schemas for the following source data warehouses to a supported target. For information about permissions, connections, and what AWS SCT can convert for use with the target database or data warehouse, see details in the following.

Topics
- Using Amazon Redshift as a source for AWS SCT (p. 101)
- Using Azure Synapse Analytics as a source for AWS SCT (p. 103)
- Using Greenplum Database as a source for AWS SCT (p. 104)
- Using Netezza as a source for AWS SCT (p. 106)
- Using Oracle Data Warehouse as a source for AWS SCT (p. 109)
- Using Snowflake as a source for AWS SCT (p. 111)
- Using Microsoft SQL Server Data Warehouse as a source for AWS SCT (p. 115)
- Using Teradata as a source for AWS SCT (p. 117)
- Using Vertica as a source for AWS SCT (p. 119)

Using Amazon Redshift as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code 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

In the example preceding, replace the <schema_name> placeholder with the name of the source schema.

Connecting to Amazon Redshift as a source

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

To connect to an Amazon Redshift source database
1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose **Amazon Redshift**, then choose **Next**.

   The **Add source** dialog box appears.

3. For **Connection name**, enter a name for your database. AWS SCT displays this name in the tree in the left panel.

4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

     For information about using database credentials from Secrets Manager, see **Using AWS Secrets Manager (p. 39)**.

   - To enter the Amazon Redshift source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the Amazon Redshift database.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Enter the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. 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>Choose this option 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>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. For this location to appear here, make sure to add it in <strong>Global settings</strong>.</td>
</tr>
<tr>
<td></td>
<td>For more information about SSL support for Amazon Redshift, see <strong>Configure security options for connections</strong>.</td>
</tr>
<tr>
<td><strong>Store password</strong></td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td><strong>Redshift driver path</strong></td>
<td>Enter 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>
</tbody>
</table>
Using Azure Synapse Analytics as a source

You can use AWS SCT to convert schemas, code objects, and application code from Azure Synapse Analytics to Amazon Redshift.

Privileges for Azure Synapse Analytics as a source

The privileges required for Azure Synapse Analytics data warehouse as a source are listed following:

- VIEW DEFINITION
- VIEW DATABASE STATE

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

Connecting to Azure Synapse Analytics as a source

Use the following procedure to connect to your Azure Synapse Analytics source data warehouse with the AWS Schema Conversion Tool.

To connect to an Azure Synapse Analytics source data warehouse

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Azure Synapse Analytics, then choose Next.

   The Add source dialog box appears.
3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
     2. Choose Populate to automatically fill in all values in the database connection dialog box from Secrets Manager.
   
     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).
   - To enter the Azure Synapse Analytics source data warehouse connection information manually, use the instructions in the following table.

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<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. 11).</td>
</tr>
</tbody>
</table>

5. Choose Test Connection to verify that AWS SCT can connect to your source database.
6. Choose Connect to connect to your source database.
### For this parameter | Do this
--- | ---
**Server name** | Enter the Domain Name Service (DNS) name or IP address of your source database server.

**SQL pool** | Enter the name of the Azure SQL pool.

**User name** and **Password** | Enter the user name and password to connect to your source database server.

AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.

**Use SSL** | Choose this option to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the **SSL** tab:
- **Trust server certificate**: Choose this option to trust the server certificate.
- **Trust store**: A trust store that you set up in the Global settings.

**Store password** | AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.

---

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.

### Using Greenplum Database as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from Greenplum Database to Amazon Redshift.

### Privileges for Greenplum Database as a source

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

- CONNECT ON DATABASE `<database_name>`
- USAGE ON SCHEMA `<schema_name>`
- SELECT ON `<schema_name>`.`<table_name>`
- SELECT ON SEQUENCE `<schema_name>`.`<sequence_name>`

In the example preceding, replace placeholders as following:

- Replace `<database_name>` with the name of the source database.
- Replace `<schema_name>` with the name of the source schema.
• Replace `table_name` with the name of the source table.
• Replace `sequence_name` with the name of the sequence name.

**Connecting to Greenplum Database as a source**

Use the following procedure to connect to your Greenplum source database with AWS SCT.

**To connect to a Greenplum source database**

1. In the AWS Schema Conversion Tool, choose **Add source**.
2. Choose **SAP ASE**, then choose **Next**.

   The **Add source** dialog box appears.
3. For **Connection name**, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   • To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).
   • To enter the Greenplum source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the Greenplum database.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Enter the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn’t store the password by default. 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>Choose this option 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>• <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>
</tbody>
</table>
Using Netezza as a source

You can use AWS SCT to convert schemas, code objects, and application code 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 <schema_name>
- list on <database_name>.all.table
- list on <database_name>.all.external table
- list on <database_name>.all.view
- list on <database_name>.all.materialized view
- list on <database_name>.all.procedure
- list on <database_name>.all.sequence
- list on <database_name>.all.function
- list on <database_name>.all.aggregate

In the example preceding, replace placeholders as following:

- Replace database_name with the name of the source database.
- Replace schema_name with the name of the source schema.

AWS SCT requires access to the following system tables and views. You can grant access to these objects instead of granting access to system.definition_schema.system view and system.definition_schema.system tables in the preceding list.

- select on system.definition_schema._t_aggregate
• select on system.definition_schema._t_class
• select on system.definition_schema._t_constraint
• select on system.definition_schema._t_const_reattr
• select on system.definition_schema._t_database
• select on system.definition_schema._t_grpobj_priv
• select on system.definition_schema._t_grpusr
• select on system.definition_schema._t_hist_config
• select on system.definition_schema._t_object
• select on system.definition_schema._t_object_classes
• select on system.definition_schema._t_proc
• select on system.definition_schema._t_type
• select on system.definition_schema._t_user
• select on system.definition_schema._t_usrobj_priv
• select on system.definition_schema._vt_sequence
• select on system.definition_schema._v_aggregate
• select on system.definition_schema._v_constraint_depends
• select on system.definition_schema._v_database
• select on system.definition_schema._v_datatype
• select on system.definition_schema._v_dslice
• select on system.definition_schema._v_function
• select on system.definition_schema._v_group
• select on system.definition_schema._v_obj_relation
• select on system.definition_schema._v_obj_relation_xdb
• select on system.definition_schema._v_procedure
• select on system.definition_schema._v_relation_column
• select on system.definition_schema._v_relation_keydata
• select on system.definition_schema._v_relobjclasses
• select on system.definition_schema._v_schema_xdb
• select on system.definition_schema._v_sequence
• select on system.definition_schema._v_synonym
• select on system.definition_schema._v_system_info
• select on system.definition_schema._v_sys_constraint
• select on system.definition_schema._v_sys_object_dslice_info
• select on system.definition_schema._v_sys_user
• select on system.definition_schema._v_table
• select on system.definition_schema._v_table_constraint
• select on system.definition_schema._v_table_dist_map
• select on system.definition_schema._v_table_organize_column
• select on system.definition_schema._v_table_storage_stat
• select on system.definition_schema._v_user
• select on system.definition_schema._v_view
• select on system.information_schema._v_relation_column
• select on system.information_schema._v_table
• select on $hist_column_access_*
Connecting to Netezza as a source

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

To connect to a Netezza source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Netezza, then choose Next.

The Add source dialog box appears.

3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:

   • To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
     2. Choose Populate to automatically fill in all values in the database connection dialog box from Secrets Manager.

   For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).

   • To enter the Netezza source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Enter the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. 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. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td>Netezza driver path</td>
<td>Enter 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. 11).</td>
</tr>
</tbody>
</table>
5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.

### Using Oracle Data Warehouse as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from Oracle Data Warehouse to Amazon Redshift or Amazon Redshift and AWS Glue used in combination.

### 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 Oracle Data Warehouse as a source

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

**To connect to an Oracle Data Warehouse source database**

1. In the AWS Schema Conversion Tool, choose **Add source**.
2. Choose **Oracle**, then choose **Next**.

   The **Add source** dialog box appears.

3. For **Connection name**, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.
   - For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).
   - To enter the Oracle source data warehouse connection information manually, use the instructions in the following table.

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</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 Domain Name System (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>For this parameter</td>
<td>Do this</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Oracle SID</strong></td>
<td>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><strong>Service Name</strong></td>
<td>The DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server name</strong></td>
<td>The port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>The name of the Oracle service to connect to.</td>
</tr>
<tr>
<td><strong>TNS alias</strong></td>
<td>The path to the file that contains the Transparent Network Substrate (TNS) name connection information.</td>
</tr>
<tr>
<td><strong>TNS file path</strong></td>
<td>The TNS alias from this file to use to connect to the source database.</td>
</tr>
<tr>
<td><strong>TNS connect identifier</strong></td>
<td>The identifier for the registered TNS connection information.</td>
</tr>
</tbody>
</table>

**User name and Password**

Enter the user name and password to connect to your source database server.

AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.

**Use SSL**

Choose this option to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:

- **SSL authentication**: Select this option to use SSL authentication for the connection.
- **Trust store**: The location of a trust store containing certificates.
- **Key store**: 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.

**Store password**

AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.
Using Snowflake as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from Snowflake to Amazon Redshift.

Privileges for Snowflake as a source database

You can create a role with privileges and grant this role the name of a user by using the SECURITYADMIN role and the SECURITYADMIN session context.

The example following creates minimal privileges and grants them to the min_privs user.

```sql
create role role_name;
grant role role_name to role sysadmin;
grant usage on database db_name to role role_name;
grant usage on schema db_name.schema_name to role role_name;
grant usage on warehouse datawarehouse_name to role role_name;
grant monitor on database db_name to role role_name;
grant monitor on warehouse datawarehouse_name to role role_name;
grant select on all tables in schema db_name.schema_name to role role_name;
grant select on future tables in schema db_name.schema_name to role role_name;
grant select on all views in schema db_name.schema_name to role role_name;
grant select on future views in schema db_name.schema_name to role role_name;
grant select on all external tables in schema db_name.schema_name to role role_name;
grant usage on all sequences in schema db_name.schema_name to role role_name;
grant usage on future sequences in schema db_name.schema_name to role role_name;
grant usage on all functions in schema db_name.schema_name to role role_name;
grant usage on future functions in schema db_name.schema_name to role role_name;
grant usage on all procedures in schema db_name.schema_name to role role_name;
grant usage on future procedures in schema db_name.schema_name to role role_name;
create user min_privs password='real_user_password'
DEFAULT_ROLE = role_name DEFAULT_WAREHOUSE = 'datawarehouse_name';
grant role role_name to user min_privs;
```

In the example preceding, replace placeholders as following:

- Replace **role_name** with the name of a role with read-only privileges.
- Replace **db_name** with the name of the source database.
- Replace **schema_name** with the name of the source schema.
- Replace **datawarehouse_name** with the name of a required data warehouse.
- Replace **min_privs** with the name of a user that has minimal privileges.
The `DEFAULT_ROLE` and `DEFAULT_WAREHOUSE` parameters are key-sensitive.

## Configuring secure access to Amazon S3

Security and access management policies for an Amazon S3 bucket allow Snowflake to access, read data from, and write data to the S3 bucket. You can configure secure access to a private Amazon S3 bucket using the Snowflake `STORAGE_INTEGRATION` object type. A Snowflake storage integration object delegates authentication responsibility to a Snowflake identity and access management entity.

For more information, see Configuring a Snowflake Storage Integration to Access Amazon S3 in the Snowflake documentation.

## Connecting to Snowflake as a source

Use the following procedure to connect to your source database with the AWS Schema Conversion Tool.

**To connect to an Snowflake source database**

1. In the AWS Schema Conversion Tool, choose **Add source**.
2. Choose **Snowflake**, then choose **Next**.

   The **Add source** dialog box appears.

3. For **Connection name**, enter a name for your database. AWS SCT displays this name in the tree in the left panel.

4. Use database credentials from AWS Secrets Manager or enter them manually:

   - To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).

   - To enter the Snowflake source data warehouse connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the Snowflake database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Enter the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>AWS SCT stores your password in an encrypted format only if you explicitly request it.</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Choose 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>Private key path</strong>: The location of a private key.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Passphrase</strong>: The passphrase for the private key.</td>
</tr>
</tbody>
</table>
Using Snowflake as a source

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.

**Limitations for Snowflake as a source**

The following are limitations when using Snowflake as a source for AWS SCT:

- Object identifiers must be unique within the context of the object type and the parent object:
  - **Database**
    - Schema identifiers must be unique within a database.
  - **Schemas**
    - Objects identifiers such as for tables and views must be unique within a schema.
  - **Tables/Views**
    - Column identifiers must be unique within a table.
    - The maximum number of tables for large and xlarge cluster node types is 9,900. For 8xlarge cluster node types, the maximum number of tables is 100,000. The limit includes temporary tables, both user-defined and created by Amazon Redshift during query processing or system maintenance. For more information, see Amazon Redshift quotas in the Amazon Redshift Cluster Management Guide.
    - For stored procedures, the maximum number of input and output arguments is 32.

**Source data types for Snowflake**

Following, you can find the Snowflake source data types that are supported when using AWS SCT and the default mapping to an Amazon Redshift target.

<table>
<thead>
<tr>
<th>Snowflake data types</th>
<th>Amazon Redshift data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER</td>
<td>NUMERIC(38)</td>
</tr>
<tr>
<td>NUMBER(p)</td>
<td>If p is =&lt; 4, then SMALLINT</td>
</tr>
<tr>
<td></td>
<td>If p is =&gt; 5 and =&lt; 9, then INTEGER</td>
</tr>
<tr>
<td>Snowflake data types</td>
<td>Amazon Redshift data types</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------</td>
</tr>
</tbody>
</table>
| NUMBER(p, 0)         | If p is => 10 and <= 18, then BIGINT  
If p is => 19 then NUMERIC(p) |
| NUMBER(p, s)         | If p is => 1 and <= 38, and if s is => 1 and <= 37, then NUMERIC(p,s) |
| FLOAT                | FLOAT                      |
| TEXT                 | VARCHAR(MAX)               |
| TEXT(p)              | If p is <= 65,535 then, VARCHAR(p) |
| TEXT(p)              | If p is => 65,535 and <= 16,777,216 then, VARCHAR(MAX) |
| BINARY               | VARCHAR(MAX)               |
| BINARY(p)            | VARCHAR(p)                 |
| BINARY(p)            | VARCHAR(MAX)               |
| BOOLEAN              | BOOLEAN                    |
| DATE                 | DATE                       |
| TIME                 | VARCHAR(18)                |

Unicode characters up to 16,777,216 bytes; up to 4 bytes per character.

Single-byte characters up to 8,388,608 bytes; 1 byte per character.

Time values between 00:00:00 and 23:59:59.999999999.
AWS Schema Conversion Tool User Guide

Using SQL Server Data Warehouse as a source

### Snowflake data types

<table>
<thead>
<tr>
<th>Snowflake data types</th>
<th>Amazon Redshift data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME(f)</td>
<td>VARCHAR(n) – 9 + dt-attr-1</td>
</tr>
<tr>
<td>Time values between 00:00:00 and 23:59:59.9(f).</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_NTZ</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>TIMESTAMP_TZ</td>
<td>TIMESTAMPTZ</td>
</tr>
</tbody>
</table>

### Using Microsoft SQL Server Data Warehouse as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from Microsoft SQL Server DW to Amazon Redshift or Amazon Redshift and AWS Glue used in combination.

### 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>

In the example preceding, replace the `<source_schema>` placeholder with the name of the source source_schema.

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

### Limitations for SQL Server Data Warehouse as a source

Using Microsoft SQL Server Parallel Data Warehouse (PDW) as a source isn't currently supported.

### Connecting to SQL Server Data Warehouse as a source

Use the following procedure to connect to your SQL Server Data Warehouse source database with the AWS Schema Conversion Tool.

**To connect to a SQL Server Data Warehouse source database**

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Microsoft SQL Server, then choose Next.
   
   The Add source dialog box appears.
3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:

   - To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).

   - To enter the Microsoft SQL Server source data warehouse connection information manually, use the instructions in the following table.

     | For this parameter | Do this |
     |-------------------|---------|
     | Server name       | Enter the Domain Name Service (DNS) name or IP address of your source database server. |
     | Server port       | Enter the port used to connect to your source database server. |
     | Instance name     | Enter the instance name for the SQL Server data warehouse. |
     | User name and Password | Enter the user name and password to connect to your source database server. |
     |                   | AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed. |
     | Use SSL           | Choose this option to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the SSL tab: |
     |                   | • **Trust server certificate**: Select this option to trust the server certificate. |
     |                   | • **Trust store**: A trust store that you set up in the **Global settings**. |
     | Store password    | AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password. |
     | SQL Server driver path | Enter 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. 11). |

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.

6. Choose **Connect** to connect to your source database.
Using Teradata as a source for AWS SCT

You can use AWS SCT to convert schemas, code objects, and application code from Teradata to Amazon Redshift or Amazon Redshift and AWS Glue used in combination.

Privileges for Teradata as a source

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

- SELECT ON DBC
- SELECT ON SYSUDTLIB
- SELECT ON SYSLIB
- SELECT ON `<source_database>`
- CREATE PROCEDURE ON `<source_database>`

In the example preceding, replace the `<source_database>` placeholder with the name of the source database.

AWS SCT requires the CREATE PROCEDURE privilege to perform HELP PROCEDURE against all procedures in the source database. AWS SCT doesn't use this privilege to create any new objects in your source Teradata database.

Connecting to Teradata as a source

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

To connect to a Teradata source database

1. In the AWS Schema Conversion Tool, choose Add source.
2. Choose Teradata, then choose Next.
   
   The Add source dialog box appears.
3. For Connection name, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   
   - To use database credentials from Secrets Manager, use the following instructions:
     1. For AWS Secret, choose the name of the secret.
     2. Choose Populate to automatically fill in all values in the database connection dialog box from Secrets Manager.
     
     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).
   
   - To enter the Teradata source database connection information manually, use the instructions in the following table.

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection name</td>
<td>Enter a name for your database. AWS SCT displays this name in the tree in the left panel.</td>
</tr>
<tr>
<td>Server name</td>
<td>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>For this parameter</td>
<td>Do this</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Server port</td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the Teradata database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Enter the user name and password to connect to your source database. AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. 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. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td>Encrypt data</td>
<td>Choose this option to encrypt data that you exchange with the database.</td>
</tr>
<tr>
<td>Teradata driver path</td>
<td>Enter 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. 11).</td>
</tr>
</tbody>
</table>

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.

6. Choose **Connect** 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 following procedure, the Active Directory domain is `test.local.com`. The Windows server is `DC`, and it's configured with default settings. The user account created in Active Directory is `test_ldap`, and the account uses the password `test_ldap`.

**To set up LDAP authentication for Teradata users who run Microsoft Active Directory in Windows**

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

   AuthorizationSupported="no"

   LdapServerName="DC.test.local.com"
   LdapServerPort="389"
   LdapServerRealm="test.local.com"
   LdapSystemFQDN="dc= test, dc= local, dc=com"
LdapBaseFQDN="dc=test, dc=local, dc=com"

2. Apply the changes by running the configuration as follows.

```
# cd /opt/teradata/tdgss/bin
# ./run_tdgssconfig
```

3. Test the configuration by running the following command.

```
# /opt/teradata/tdat/tdgss/14.10.03.01/bin/tdsbind -u test_ldap -w test_ldap
```

The output should be similar to the following.

```
LdapGroupBaseFQDN: dc=Test, dc=local, dc=com
LdapUserBaseFQDN: dc=Test, dc=local, dc=com
LdapSystemFQDN: dc=test, dc=local, dc=com
LdapServerName: DC.test.local.com
LdapServerPort: 389
LdapServerRealm: test.local.com
LdapClientUseTls: no
LdapClientTlsReqCert: never
LdapClientMechanism: SASL/DIGEST-MD5
LdapServiceBindRequired: no
LdapClientTlsCRLCheck: none
LdapAllowUnsafeServerConnect: yes
UseLdapConfig: no
AuthorizationSupported: no
FQDN: CN=test, CN=Users, DC=Anthem, DC=local, DC=com
AuthUser: ldap://DC.test.local.com:389/CN=test1,CN=Users,DC=test,DC=local,DC=com
DatabaseName: test
Service: tdsbind
```

4. Restart TPA using the following command.

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

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

```
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, specify this new password during connection to Teradata in LDAP mode. In DEFAULT mode, you connect to Teradata by using the LDAP user name and any password.

**Using Vertica as a source for AWS SCT**

You can use AWS SCT to convert schemas, code objects, and application code 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
• 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)>`

In the example preceding, replace placeholders as following:
• Replace `<schema_name>` with the name of the source schema.
• Replace `<procedure_name>` with the name of a source procedure. Repeat the grant for each procedure that you are converting.
• Replace `<procedure_signature>` with the comma-delimited list of procedure argument types.

**Connecting to Vertica as a source**

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

**To connect to a Vertica source database**

1. In the AWS Schema Conversion Tool, choose **Add source**.
2. Choose **Vertica**, then choose **Next**.
   The **Add source** dialog box appears.
3. For **Connection name**, enter a name for your database. AWS SCT displays this name in the tree in the left panel.
4. Use database credentials from AWS Secrets Manager or enter them manually:
   • To use database credentials from Secrets Manager, use the following instructions:
     1. For **AWS Secret**, choose the name of the secret.
     2. Choose **Populate** to automatically fill in all values in the database connection dialog box from Secrets Manager.

     For information about using database credentials from Secrets Manager, see Using AWS Secrets Manager (p. 39).
   • To enter the Vertica source database connection information manually, 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>Enter the Domain Name System (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Enter the name of the Vertica database.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Enter the user name and password to connect to yoursource database server.</td>
</tr>
<tr>
<td></td>
<td>AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. If you close your AWS SCT project and reopen it, you are</td>
</tr>
<tr>
<td>For this parameter</td>
<td>Do this</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Choose this option 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>• <strong>Verify server certificate</strong>: Choose this option to verify the server certificate by using a trust store.</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></td>
<td>• <strong>Key store</strong>: A key 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. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td><strong>Vertica driver path</strong></td>
<td>Enter the path to the driver to use to connect to the source database. For more information, see <em>Installing the required database drivers</em> (p. 8).</td>
</tr>
</tbody>
</table>

5. Choose **Test Connection** to verify that AWS SCT can connect to your source database.
6. Choose **Connect** to connect to your source database.
Creating mapping rules in AWS SCT

You can add multiple source and target databases in a single AWS SCT project. Doing this simplifies the management of projects, when you migrate multiple databases to different target platforms.

After you create a new project and add source and target databases, create mapping rules. AWS SCT requires at least one mapping rule to create a migration assessment report and convert database schemas.

A mapping rule describes a source-target pair that includes a source database schema or source database and a target database platform. You can create multiple mapping rules in a single AWS SCT project. Use mapping rules to convert every source database schema to the right target database platform.

You can create mapping rules only for supported database conversion pairs. For the list of supported conversion pairs, see Sources for AWS SCT (p. 43).

If you open a project saved in AWS SCT version 1.0.655 or before, AWS SCT automatically creates mapping rules for all source database schemas to the target database platform. To add other target database platforms, delete existing mapping rules and then create new mapping rules.

Topics
• Adding a new mapping rule (p. 122)
• Managing mapping rules (p. 122)
• Using virtual targets (p. 123)
• Limitations to using multiple servers in a single AWS SCT project (p. 124)

Adding a new mapping rule

You can create multiple mapping rules in a single project. AWS SCT saves mapping rules as part of your project. With your project open, use the following procedure to add a new mapping rule.

To create mapping rules
2. In the left panel, choose a schema or a database to add to the mapping rule.
3. In the right panel, choose a target database platform for the selected source schema or database.
4. Choose Create mapping.

AWS SCT adds this new mapping rule to the Server mappings list.

Add mapping rules for all conversion pairs. To create an assessment report or convert database schemas, choose Main view on the View menu.

AWS SCT highlights in bold all schema objects that are part of a mapping rule.

Managing mapping rules

You can filter or delete existing mapping rules, and add a new mapping rule in your AWS Schema Conversion Tool (AWS SCT) project.
When you create a mapping rule for the whole source database, AWS SCT creates one mapping rule for each source database schema. For projects that involve dozens of schemas or even databases, it may be hard to understand, which target is used for a certain schema. To quickly find a mapping rule for your schema, use one or several of the following filter options in AWS SCT.

To filter mapping rules

2. For Source servers, choose the source database.
   
   The filter default is All, which means that AWS SCT displays mapping rules for all source databases.
3. For Source schema, enter the source schema name. Use the percent (%) as a wildcard to replace any number of any symbols in the schema name.
   
   The filter default is the % wildcard, which means that AWS SCT displays mapping rules for all source database schema names.
4. For Has migration rules, choose Yes to display mapping rules for which the data migration rules are created. Choose No to display mapping rules which don’t have data migration rules. For more information, see Creating data migration rules in AWS SCT (p. 230).
   
   The filter default is All, which means that AWS SCT displays all mapping rules.
5. For Target servers, choose the target database.
   
   The filter default is All, which means that AWS SCT displays mapping rules for all target databases.

With your project open, use the following procedure to delete a mapping rule. For more information on adding mapping rules, see Adding a new mapping rule (p. 122).

To delete mapping rules

2. For Server mappings, choose the mapping rules to delete.
3. Choose Delete selected mappings.

AWS SCT deletes the selected mapping rules.

Using virtual targets

You can see how AWS SCT converts your source database schema to any supported target database platform. To do so, you don’t need to connect to an existing target database. Instead, you can use a virtual target database platform in a mapping rule.

AWS SCT supports the following virtual target database platforms:

- Amazon Redshift
- Amazon Redshift and AWS Glue
- Aurora MySQL
- Aurora PostgreSQL
- Babelfish for Aurora PostgreSQL
- MariaDB
- Microsoft SQL Server
- MySQL
- Oracle
• PostgreSQL

If you use Babelfish for Aurora PostgreSQL as a target database platform, you can only create a database migration assessment report. For more information, see the section called "Migration assessment reports" (p. 125).

If you use a virtual target database platform, you can save converted code to a file. For more information, see the section called “Saving your converted schema” (p. 153).

Limitations to using multiple servers in a single AWS SCT project

The following limitations apply when converting schemas using multiple servers in a single AWS SCT project:

• You can add the same server to a project only once.
• You can't map server schemas to a specific target schema, only to a target server. AWS SCT creates the target schema during conversion.
• You can't map lower-level source objects to the target server.
• You can map one source schema to only one target server in a project.
• Make sure to map a schema to a target server to create an assessment report, convert schemas, or extract data.
Creating conversion reports

When you are planning a database conversion, it is helpful to create some reports to help you understand what is involved. You can create reports using AWS Schema Conversion Tool.

You can use AWS SCT to create a database migration assessment report. With this report, you get a summary of your schema conversion tasks and the details for items that can't be automatically converted to your target database. You can use this report to evaluate how much of the project can be completed by using AWS SCT, and what else you need to complete the conversion. To create an assessment report, use Create Report from the context (right-click) menu of the database in AWS SCT.

Topics

• Creating migration assessment reports with AWS SCT (p. 125)

Creating migration assessment reports with AWS SCT

An important part of the AWS Schema Conversion Tool is the report that it generates to help you convert your schema. This database migration assessment 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. You can view the report in the application or export it as a comma-separated value (CSV) or PDF file.

If you add multiple source and target databases in a single project, AWS SCT aggregates the reports for all conversion pairs into one database migration assessment report.

You can use virtual target database platforms to generate an assessment report and understand the complexity of migration to a selected database platform. In this case, you don't need to connect to your target database platform. For example, you can use Babelfish for Aurora PostgreSQL as a virtual target database platform to create a database migration assessment report. For more information on virtual target database platforms, see the section called “Virtual targets” (p. 123).

The migration assessment report includes the following:

• Executive summary
• License evaluation
• Cloud support, indicating any features in the source database not available on the target.
• Recommendations, including conversion of server objects, backup suggestions, and linked server changes

The report also includes estimates of the amount of effort that it will take to write the equivalent code for your target DB instance that can't be converted automatically.

If you use AWS SCT to migrate your existing schema to an Amazon RDS DB instance, the report can help you analyze requirements for moving to the AWS Cloud and for changing your license type.

Topics

• Creating a database migration assessment report (p. 126)
• Viewing the assessment report (p. 126)
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. On the View menu, choose Main view.
2. In the left panel that displays your source database schema, choose a schema object to create an assessment report for. To include multiple database schemas into the report, choose the parent node, for example Schemas.
3. Open the context (right-click) menu for the object, and then choose Create Report.

Viewing the assessment report

After you create an assessment report, the assessment report view opens, showing the following tabs:

- Summary
- Action Items
The **Summary** tab shows items that were automatically converted or not converted.

The **Action Items** tab shows items that couldn't be converted automatically, and recommendations on what to do about them.

**Topics**
- Assessment report summary (p. 127)
- Assessment report action items (p. 131)
- Assessment report warning message (p. 133)

**Assessment report summary**

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.
Database migration assessment report

Source database: GOLD_TEST_SS_PG-21-76.eu-west-1.compute.amazonaws.com\GOLD_TEST_SS_PG:1433
Microsoft SQL Server 2019 (RTM-CU10) (KB5001090) - 15.0.4123.1 (X64) Mar 22 2021 18:10:24
Copyright (C) 2019 Microsoft Corporation
Enterprise Edition: Core-based Licensing (64-bit) on Windows Server 2019 Datacenter 10.0 <X64> (Build 17763:) (Hypervisor)
Case sensitivity: OFF

Executive summary

We completed the analysis of your Microsoft SQL Server source database and estimate that 90% of the database storage objects and 77% of database actions are ready to be migrated to the target. Database storage objects include schemas, tables, table constraints, indexes, types, table types, sequences, synonyms and XML schema collections. Based on the source code syntax analysis, we estimate 94% (based on # of lines of code) of your code can be converted to Amazon RDS database migration actions and medium-complexity actions to complex conversion actions.

Migration guidance for database objects that could not be converted automatically can be found here

Database objects with conversion actions for Amazon RDS for SQL Server

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identified 529 (90%) database storage objects and 1,506 (98%) database code objects that can be converted to Amazon RDS database migration actions and medium-complexity actions to complex conversion actions.

We found 7 encrypted object(s).

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

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

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple actions has a higher complexity level than an object with a single complex action.

Figure: Conversion statistics for database storage objects

<table>
<thead>
<tr>
<th>Schema (4: 4/0/0/0)</th>
<th>100%</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table (323: 276/8/2/37)</td>
<td>85%</td>
<td></td>
</tr>
<tr>
<td>Constraint (157: 152/2/0/3)</td>
<td>97%</td>
<td></td>
</tr>
<tr>
<td>Index (63: 36/22/0/5)</td>
<td>57%</td>
<td>35%</td>
</tr>
<tr>
<td>Type (7: 7/0/0/0)</td>
<td>100%</td>
<td>7</td>
</tr>
<tr>
<td>Sequence (14: 7/7/0/0)</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Synonym (5: 0/0/0/5)</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Table Type (7: 7/0/0/0)</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>XML schema collection</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
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 two hours.
- **Medium** – Actions that are more complex and can be completed in two to six hours.
- **Significant** – Actions that are very complex and take more than six 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.
License evaluation

Our analysis shows that current schema uses the following Enterprise Edition features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database In-Memory</td>
<td>Oracle Database In-Memory</td>
</tr>
<tr>
<td>Materialized View Query Rewrite</td>
<td>Oracle Database employs an</td>
</tr>
<tr>
<td>Partitioning</td>
<td>Partitioning is powerful function</td>
</tr>
<tr>
<td>Oracle Advanced Security/TDE</td>
<td>Oracle Advanced Security product</td>
</tr>
</tbody>
</table>

If you choose Standard Edition as your migration target, remove dependencies on the above features.

Cloud support

Our analysis shows that your current schema uses the following features that require special consideration:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locator</td>
<td>Oracle Locator provides capabilities. Please read prerequisites and</td>
</tr>
<tr>
<td>Spatial</td>
<td>Oracle Spatial provides a SQL layer. Please read prerequisites and</td>
</tr>
<tr>
<td>Oracle XML DB</td>
<td>Oracle XML DB provides full support. Amazon RDS for Oracle support</td>
</tr>
</tbody>
</table>

If choose Amazon RDS for Oracle as your migration target, please follow the above...
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 example, after the assessment runs, detailed reports for the database/schema show you the effort required to design and implement the recommendations for converting Action items. For more information about deciding how to handle manual conversions, see Chapter 12: Handling manual conversions in AWS SCT on page 152.
Assessment report warning message

To assess the complexity of converting to another database engine, AWS SCT requires access to objects in your source database. When SCT can't perform an assessment because problems were encountered during scanning, a warning message is issued that indicates overall conversion percentage is reduced.

Following are reasons why AWS SCT might encounter problems during scanning:

- The user account connected to the database doesn't have access to all of the needed objects.
- An object cited in the schema no longer exists in the database.
- SCT is trying to assess an object that is encrypted.

For more information about SCT required security permissions and privileges for your database, see Sources for AWS SCT (p. 43) for the appropriate source database section in this guide.

Saving the assessment report

After you create a database migration assessment report (p. 126), you can save a local copy of the database migration assessment report as either a PDF file or a comma-separated value (CSV) file.

To save a database migration assessment report as a PDF file

1. In the top menu, choose View, and then choose Assessment Report view.
2. Choose the Summary tab.
3. Choose Save to PDF at upper right.

To save a database migration assessment report as a CSV file

1. In the top menu, choose View, and then choose Assessment Report view.
2. Choose the Summary tab.
3. Choose Save to CSV at upper right.

The PDF file contains both the summary and action item information, as shown in the following example.
Database objects with conversion actions for Amazon RDS PostgreSQL

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

We found 7 encrypted object(s).

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

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

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

---

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

- **Schema (4/4/0/0)**: 100% conversion rate, 4 objects.
- **Table (323/276/8/2/37)**: 85% conversion rate, 11% requires complex actions, 11% requires medium actions, 37 objects.
- **Constraint (157/152/2/0/3)**: 97% conversion rate, 3% requires complex actions, 157 objects.
- **Index (63/36/22/0/5)**: 57% conversion rate, 35% requires medium actions, 8% requires complex actions, 63 objects.
- **Type (7/7/0/0/0)**: 100% conversion rate, 7 objects.
- **Sequence (14/7/7/0/0)**: 50% conversion rate, 50% requires medium actions, 14 objects.
- **Synonym (5/0/0/5)**: 100% conversion rate, 5 objects.
- **Table Type (7/7/0/0/0)**: 100% conversion rate, 7 objects.
- **Xml schema collection (5/0/0/4/0)**: 20% conversion rate, 80% requires medium actions, 5 objects.
When you choose the **Save to CSV** option, AWS SCT creates three CSV files.

The first CSV file contains the following information about action items:

- Category
- Occurrence – The file name, line number, and position for the item
- Action item number
- Subject
- Group
- Description
- Documentation references
- Recommended action
- Estimated complexity

The second CSV file includes the *Action_Items_Summary* suffix in its name and contains the information about the number of occurrences of all action items.

In the following example, values in the **Learning curve effort** column indicate the amount of effort needed to design an approach to converting each action item. Values in the **Effort to convert an occurrence of the action item** column indicate the effort needed to convert each action item, following the designed approach. The values used to indicate the level of effort needed are based on a weighted scale, ranging from low (least) to high (most).

<table>
<thead>
<tr>
<th>Schema</th>
<th>Action item</th>
<th>Number of occurrences</th>
<th>Learning curve efforts</th>
<th>Efforts to convert</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST.dbo</td>
<td>609</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>TEST.dbo</td>
<td>681</td>
<td>2</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>TEST.dbo</td>
<td>690</td>
<td>1</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>TEST.dbo</td>
<td>794</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TEST.dbo</td>
<td>811</td>
<td>12</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>TEST.dbo</td>
<td>826</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TEST.dbo</td>
<td>844</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>TEST.dbo</td>
<td>9997</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

The third CSV file includes *Summary* in its name and contains the following summary:

- Category
- Number of objects
- Objects automatically converted
- Objects with simple actions
- Objects with medium-complexity actions
- Objects with complex actions
- Total lines of code
Creating a multiserver assessment report for database migration

To determine the best target direction for your overall environment, create a multiserver assessment report.

A multiserver assessment report evaluates multiple servers based on input that you provide for each schema definition that you want to assess. Your schema definition contains database server connection parameters and the full name of each schema. After assessing each schema, AWS SCT produces a summary, aggregated assessment report for database migration across your multiple servers. This report shows the estimated complexity for each possible migration target.

You can use AWS SCT to create a multiserver assessment report for the following source and target databases.

<table>
<thead>
<tr>
<th>Source database</th>
<th>Target database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Redshift</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Greenplum</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>IBM Db2 for z/OS</td>
<td>Amazon Aurora MySQL-Compatible Edition (Aurora MySQL), Amazon Aurora PostgreSQL-Compatible Edition (Aurora PostgreSQL), MySQL, PostgreSQL</td>
</tr>
<tr>
<td>IBM Db2 LUW</td>
<td>Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>Microsoft Azure SQL</td>
<td>Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora MySQL, Aurora PostgreSQL, Amazon Redshift, Babelfish for Aurora PostgreSQL, MariaDB, Microsoft SQL Server, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>MySQL</td>
<td>Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora MySQL, Aurora PostgreSQL, Amazon Redshift, MariaDB, MySQL, Oracle, PostgreSQL</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
</tr>
</tbody>
</table>
Performing a multiserver assessment

Use the following procedure to perform a multiserver assessment with AWS SCT. You don't need to create a new project in AWS SCT to perform a multiserver assessment. Before you get started, make sure that you have prepared a comma-separated value (CSV) file with database connection parameters. Also, make sure that you have installed all required database drivers and set the location of the drivers in the AWS SCT settings. For more information, see Installing the required database drivers (p. 8).

To perform a multiserver assessment and create an aggregated summary report


2. Choose Download a connections file example to download an empty template of a CSV file with database connection parameters.

3. Enter values for Project name, Location (to store reports), and Connections file (a CSV file).

4. Choose Create AWS SCT projects for each source database to automatically create migration projects after generating the assessment report.

5. With the Create AWS SCT projects for each source database turned on, you can choose Add mapping rules to these projects and save conversion statistics for offline use. In this case, AWS SCT will add mapping rules to each project and save the source database metadata in the project. For more information, see Running AWS SCT in an offline mode (p. 18).


A progress bar appears indicating the pace of database assessment. The number of target engines can affect the assessment runtime.

7. Choose Yes if the following message is displayed: Full analysis of all Database servers may take some time. Do you want to proceed?

When the multiserver assessment report is done, a screen appears indicating so.

8. Choose Open Report to view the aggregated summary assessment report.

Preparing an input CSV file

To provide connection parameters as input for multiserver assessment report, use a CSV file as shown in the following example.
Creating a multiserver assessment report

The example preceding uses a semicolon to separate the two schema names for the Sales database. It also uses a semicolon to separate the two target database migration platforms for the Sales database.

Also, the example preceding uses AWS Secrets Manager to connect to the Customers database and Windows Authentication to connect to the HR database.

You can create a new CSV file or download a template for a CSV file from AWS SCT and fill in the required information. Make sure that the first row of your CSV file includes the same column names as shown in the preceding example.

To download a template of the input CSV file

1. Start AWS SCT.
2. Choose File, then choose New multiserver assessment.
3. Choose Download a connections file example.

Make sure that your CSV file includes the following values, provided by the template:

- **Name** – The text label that helps identify your database. AWS SCT displays this text label in the assessment report.
- **Description** – An optional value, where you can provide additional information about the database.
- **Secret Manager Key** – The name of the secret that stores your database credentials in the AWS Secrets Manager. To use Secrets Manager, make sure that you store AWS profiles in AWS SCT. For more information, see Using AWS Secrets Manager (p. 39).
- **Server IP** – The Domain Name Service (DNS) name or IP address of your source database server.
- **Port** – The port used to connect to your source database server.
- **Service Name** – If you use a service name to connect to your Oracle database, the name of the Oracle service to connect to.
- **SID** – The database name. For Oracle databases, use the Oracle System ID (SID).
- **Source Engine** – The type of your source database. Use one of the following values:
  - **AZURE_MSSQL** for a Microsoft Azure SQL database.
  - **DB2ZOS** for an IBM Db2 for z/OS database.
  - **DB2LUW** for an IBM Db2 LUW database.
  - **GREENPLUM** for a Greenplum database.
  - **MSSQL** for a Microsoft SQL Server database.
  - **MYSQL** for a MySQL database.
  - **NETEZZA** for a Netezza database.
  - **ORACLE** for an Oracle database.
  - **POSTGRESQL** for a PostgreSQL database.
  - **REDSHIFT** for an Amazon Redshift database.
  - **SNOWFLAKE** for a Snowflake database.
  - **SYBASE_ASE** for an SAP ASE database.
• **TERADATA** for a Teradata database.
• **VERTICA** for a Vertica database.
• **Schema Names** – The names of the database schemas to include in the assessment report.

For Microsoft Azure SQL, Microsoft SQL Server, Netezza, SAP ASE, and Snowflake, use the following format of the schema name:

```
db_name.schema_name
```

Replace `db_name` with the name of the source database.

Replace `schema_name` with the name of the source schema.

Enclose database or schema names that include a dot in double quotation marks as shown following:

```
"database.name"."schema.name"
```

Separate multiple schema names by using semicolons as shown following: Schema1;Schema2.

The database and schema names are case-sensitive.

Use the percent (%) as a wildcard to replace any number of any symbols in the database or schema name. The example preceding uses the percent (%) as a wildcard to include all schemas from the employees database in the assessment report.

• **Use Windows Authentication** – If you use Windows Authentication to connect to your Microsoft SQL Server database, enter `true`. For more information, see Using Windows Authentication when using Microsoft SQL Server as a source (p. 71).

• **Login** – The user name to connect to your source database server.

• **Password** – The password to connect to your source database server.

• **Use SSL** – If you use Secure Sockets Layer (SSL) to connect to your source database, enter `true`.

• **Trust store** – The trust store to use for your SSL connection.

• **Key store** – The key store to use for your SSL connection.

• **SSL authentication** – If you use SSL authentication by certificate, enter `true`.

• **Target Engines** – The target database platforms. Use the following values to specify one or more targets in the assessment report:

  • **AURORA_MYSQL** for an Aurora MySQL-Compatible database.
  • **AURORA_POSTGRESQL** for an Aurora PostgreSQL-Compatible database.
  • **BABELFISH** for a Babelfish for Aurora PostgreSQL database.
  • **MARIA_DB** for a MariaDB database.
  • **MSSQL** for a Microsoft SQL Server database.
  • **MYSQL** for a MySQL database.
  • **ORACLE** for an Oracle database.
  • **POSTGRESQL** for a PostgreSQL database.
  • **REDSHIFT** for an Amazon Redshift database.

Separate multiple targets by using semicolons like this: `MYSQL;MARIA_DB`. The number of targets affects the time it takes to run the assessment.

### Locating and viewing reports

The multiserver assessment generates two types of reports:

• An aggregated report of all source databases.

---

**Version 1.0**

139
Creating a multiserver assessment report

- A detailed assessment report of target databases for each schema name in a source database.

Reports are stored in the directory that you chose for Location in the New multiserver assessment dialog box.

To access the detailed reports, you can navigate the subdirectories, which are organized by source database, schema name, and target database engine.

Aggregated reports show information in four columns about conversion complexity of a target database. The columns include information about conversion of code objects, storage objects, syntax elements, and conversion complexity.

The following example shows information for conversion of two Oracle database schemas to Amazon RDS for PostgreSQL.

The same four columns are appended to the reports for each additional target database engine specified.

For details on how to read this information, see following.

Output for an aggregated assessment report

The aggregated multiserver database migration assessment report in AWS Schema Conversion Tool is a CSV file with the following columns:

- Server IP
- Name
- Description
- Schema Name
- Code object conversion % for target_database
- Storage object conversion % for target_database
- Syntax elements conversion % for target_database
- Conversion complexity for target_database

To gather information, AWS SCT runs full assessment reports and then aggregates reports by schemas.

In the report, the following three fields show the percentage of possible automatic conversion based on the assessment:

Code object conversion %

The percentage of code objects in the schema that AWS SCT can convert automatically or with minimal change. Code objects include procedures, functions, views, and similar.

Storage object conversion %

The percentage of storage objects that SCT can convert automatically or with minimal change. Storage objects include tables, indexes, constraints, and similar.

Syntax elements conversion %

The percentage of syntax elements that SCT can convert automatically. Syntax elements include SELECT, FROM, DELETE, and JOIN clauses, and similar.
The conversion complexity calculation is based on the notion of action items. An action item reflects a type of problem found in source code that you need to fix manually during migration to a particular target. An action item can have multiple occurrences.

A weighted scale identifies the level of complexity for performing a migration. The number 1 represents the lowest level of complexity, and the number 10 represents the highest level of complexity.
Converting database schemas using AWS SCT

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 to consider before you do the conversion.

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

- You can 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.

**Topics**
- Creating migration rules in AWS SCT (p. 143)
- Converting your schema by using AWS SCT (p. 146)
- Handling manual conversions in AWS SCT (p. 152)
- Updating and refreshing your converted schema in AWS SCT (p. 152)
- Saving and applying your converted schema in AWS SCT (p. 153)
- Comparing database schemas (p. 155)
- Finding related transformed objects (p. 156)

AWS SCT supports the following online transaction processing (OLTP) conversions.

<table>
<thead>
<tr>
<th>Source database</th>
<th>Target database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>Microsoft SQL Server (version 2008 R2 and later)</td>
<td>Aurora MySQL, Aurora PostgreSQL, MariaDB 10.5, Microsoft SQL Server, MySQL, PostgreSQL</td>
</tr>
</tbody>
</table>
### Creating migration rules in AWS SCT

Before you convert your schema with AWS SCT, you can set up migration rules. *Migration rules* can do such things as change the data type of columns, move objects from one schema to another, and change the names of objects. For example, suppose that 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 migration rules for different source and target database engines.

You can create migration rules that perform the following tasks:

<table>
<thead>
<tr>
<th>Source database</th>
<th>Target database</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL (version 5.5 and later)</td>
<td>Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td></td>
<td>You can migrate schema and data from MySQL to an Aurora MySQL DB cluster without using AWS SCT. For more information, see <a href="#">Migrating data to an Amazon Aurora DB cluster</a>.</td>
</tr>
<tr>
<td>Oracle (version 10.2 and later)</td>
<td>Aurora MySQL, Aurora PostgreSQL, MariaDB 10.5, MySQL, Oracle, PostgreSQL</td>
</tr>
<tr>
<td>PostgreSQL (version 9.1 and later)</td>
<td>Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>SAP ASE (12.5, 15.0, 15.5, 15.7, and 16.0)</td>
<td>Aurora MySQL, Aurora PostgreSQL, MariaDB 10.5, MySQL, PostgreSQL</td>
</tr>
</tbody>
</table>

For information about converting a data warehouse schema, see [Converting data warehouse schemas to Amazon Redshift using AWS SCT](#) (p. 157).

To convert your database schema to Amazon RDS, you take the following high-level steps:

- **Creating migration rules in AWS SCT (p. 143)** – 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.
- **Converting your schema by using AWS SCT (p. 146)** – AWS SCT creates a local version of the converted schema for you to review, but it doesn’t apply it to your target DB instance until you are ready.
- **Creating migration assessment reports with AWS SCT (p. 125)** – AWS SCT creates 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 create a schema in your Amazon RDS DB instance that is compatible with your source database.
- **Handling manual conversions in AWS SCT (p. 152)** – 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 Amazon RDS DB instance.
- **Updating and refreshing your converted schema in AWS SCT (p. 152)** – You can update your AWS SCT project with the most recent schema from your source database.
- **Saving and applying your converted schema in AWS SCT (p. 153)** – When you are ready, have AWS SCT apply the converted schema in your local project to your target Amazon RDS DB instance.
• Change data type
• Move objects
• Rename objects
• Add, remove, or replace a prefix
• Add, remove, or replace a suffix

You can create migration rules for the following objects:

• Database
• Schema
• Table
• Column

Creating migration rules

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

To create migration rules

2. In Server mappings, choose a pair of source and target servers.
4. Choose Add new rule. A new row is added to the list of rules.
5. Configure your rule:
   a. For Name, enter a name for your rule.
   b. For For, choose the type of object that the rule applies to.
   c. For where, enter a filter to apply to objects before applying the migration 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 migration rule that you want to create.

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

6. After you have configured your migration rule, choose Save to save your rule. You can also choose Cancel to cancel your changes.
Transformation rules affect how the converted objects to be named or transformed. For example, you can rename a schema or table, add or remove prefixes, and it is also possible to use % as a wildcard. The order in which the rules are applied is important. Default transformation rules are always at the top of the list and can be easily moved or deleted.

The rules can be exported to a file for later use in the DMS, but please ensure the correct format is used.

Note, every rule might have to following status along with the corresponding actions:
- Successfully created enabled rule
- Rule with incorrect data entered

**Transformation rule:** For **tables** where database name is like '%'

- **Name:** Transformation rule
- **For:** table
- **where database name like:** %
- **Actions:** add prefix

[Export script for DMS] [Import script into SCT]
7. After you are done adding, editing, and deleting rules, choose Save All to save all your changes.
8. Choose Close to close the Transformation rules dialog box.

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

Exporting migration rules

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

To export migration rules

1. In the AWS Schema Conversion Tool, choose Mapping View on the View menu.
2. In Migration rules, choose a migration rule and then choose Modify migration rule.
3. Choose Export script for AWS DMS.
4. Browse to the location where you want to save your script, and then choose Save. Your migration rules are saved as a JSON script that can be consumed by AWS DMS.

Converting your schema by using AWS SCT

After you have connected your project to both your source database and your target Amazon RDS DB instance, your AWS Schema Conversion Tool 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, suppose that you choose a table to convert. If so, AWS SCT generates the schema for the table, 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.
AWS Schema Conversion Tool User Guide
Converting schema

Connected. Click to disconnect.

- Servers
  - SQL Server - ec2-52-17-21-76.eu-west-1.cce
  - Databases [12]
    - AdventureWorks2012_CS
    - alfresco
    - GOLD_TEST_SS_PG
    - LARGE_DB_SS
      - Schemas [2]
      - dbo
        - Tables [4]
  - Account
    - Currency
    - Customer
    - CustomerStateExt
      - Graph Tables
      - External Tables
    - Views [2]
    - Procedures [3]
    - SQL scalar functions [2]
    - SQL table-valued functions [1]
    - SQL inline functions [1]
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 AWS SCT (p. 152).

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 migration assessment reports with AWS SCT (p. 125).

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. 154).

**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 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 source DB instance.
Handling manual conversions in AWS SCT

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 a 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 AWS SCT (p. 153).

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 re-architect 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 AWS SCT

You can update both the source schema and the target schema in your AWS Schema Conversion Tool project.
Saving and applying your schema

- **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**.

**Note**

When you refresh your schema, AWS SCT loads metadata only as it is needed. To fully load all of your database’s schema, open the context (right-click) menu for your schema, and choose **Load schema**. For example, you can use this option to load metadata for your database all at once, and then work offline.

---

Saving and applying your converted schema in AWS SCT

When the AWS Schema Conversion Tool generates converted schema (as shown in Converting your schema by using AWS SCT (p. 146)), 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 migration assessment reports with AWS SCT (p. 125).

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**

1. Choose your schema and open the context (right-click) menu.
2. Choose **Save as SQL**.
3. Enter the name of the file and choose **Save**.
4. Save your converted schema using one of the following options:
   - **Single file**
   - **Single file per stage**
   - **Single file per statement**

**To choose the format of the SQL script**

1. On the **Settings** menu, choose **Project settings**.
2. Choose **Save scripts**.
3. For **Vendor**, choose the database platform.
4. For **Save SQL scripts to**, choose how you want to save your database schema script.
5. Choose **OK** to save the settings.

### 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.

![Apply to database](image)

### 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:

- IBM Db2 LUW: `aws_db2_ext`
- Microsoft SQL Server: `aws_sqlserver_ext`
- MySQL: `aws_mysql_ext`
- Oracle: `aws_oracle_ext`
- PostgreSQL: `aws_postgresql_ext`
- SAP ASE: `aws_sapase_ext`

For more information, see **Using the AWS Lambda functions from the AWS SCT extension pack** (p. 298).
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 PostgreSQL-Compatible Edition, 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 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 menu (right-click) 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.
The AWS Schema Conversion Tool 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.

- Amazon Redshift
- Azure Synapse Analytics (version 10)
- Greenplum Database (version 4.3 and later)
- Microsoft SQL Server (version 2008 and later)
- Netezza (version 7.0.3 and later)
- Oracle (version 10.2 and later)
- Snowflake (version 3)
- Teradata (version 13 and later)
- Vertica (version 7.2 and later)

For information about converting an online transaction processing (OLTP) database schema, see Converting database schemas using AWS SCT (p. 142).

To convert a data warehouse schema, take the following steps:

1. Specify the optimization strategy and rules, and specify the migration rules 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 migration rules in Settings. For more information on optimization strategies, see Choosing optimization strategies and rules for use with AWS SCT (p. 158). For more information about migration rules, see Creating migration rules in AWS SCT (p. 160).

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 AWS SCT (p. 159).

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 migration assessment reports with AWS SCT (p. 125).

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 using AWS SCT (p. 163).

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 AWS SCT (p. 169).
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 AWS SCT (p. 176). For more information about updating your source schema, see Updating and refreshing your converted schema in AWS SCT (p. 176).

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 AWS SCT (p. 177).

Choosing optimization strategies and rules for use with AWS SCT

To optimize how the AWS Schema Conversion Tool 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 AWS SCT (p. 159).
   - 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

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 AWS SCT (p. 169).
Collecting or uploading statistics for AWS SCT

To optimize how the AWS Schema Conversion Tool 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.
Creating migration rules

Before you convert your schema with AWS SCT, you can set up migration rules. Migration rules can do such things as change the data type of columns, move objects from one schema to another, and change the names of objects. For example, suppose that 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 create migration rules only for different source and target database engines.

You can create migration rules that perform the following tasks:

- Change data type
- Move objects
Creating migration rules

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

To create migration rules

2. In Server mappings, choose a pair of source and target servers.
4. Choose Add new rule. A new row is added to the list of rules.
5. Configure your rule:
   a. For Name, enter a name for your rule.
   b. For For, choose the type of object that the rule applies to.
   c. For where, enter a filter to apply to objects before applying the migration 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 migration rule that you want to create.
   e. Depending on the rule type, enter one or two additional values. For example, to rename an object, enter the new name of the object. To replace a prefix, enter the old prefix and the new prefix.
6. After you have configured your migration rule, choose Save to save your rule. You can also choose Cancel to cancel your changes.
Transformation rules affect how the converted objects to be named or structured. For example, you can rename a schema or table, add or remove prefixes, or use wildcards. It's possible to use % as a wildcard. The order in which the rules are applied is determined by the list order. Default transformation rules are always at the top of the list and can be modified or removed.

The rules can be exported to a file for later use in the DMS, but please note:

- **Successfully created enabled rule**
- **Rule with incorrect data entered**

**Transformation rule:** For **tables** where database name is like '%'

- **Name:** Transformation rule
- **For:** table
- **where database name like:** %
- **Actions:** add prefix

**Export script for DMS**  **Import script into SCT**
7. After you are done adding, editing, and deleting rules, choose **Save All** to save all your changes.
8. Choose **Close** to close the Transformation rules dialog box.

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

### Exporting migration 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 migration rules to AWS DMS. For more information about tasks, see [Working with AWS Database Migration Service replication tasks](#).

To export migration rules

1. In the AWS Schema Conversion Tool, choose **Mapping View** on the **View** menu.
2. In **Migration rules**, choose a migration rule and then choose **Modify migration rule**.
3. Choose **Export script for AWS DMS**.
4. Browse to the location where you want to save your script, and then choose **Save**. Your migration rules are saved as a JSON script that can be consumed by AWS DMS.

### Converting your schema using AWS SCT

After you have connected your project to both your source database and your target database, your AWS Schema Conversion Tool 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 AWS SCT (p. 176).

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 migration assessment reports with AWS SCT (p. 125).

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. 178).

## 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 AWS SCT

After you convert your schema with the AWS Schema Conversion Tool, 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.

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.

If the choices for the key don't look like what you expected, you can edit your optimization strategies, and then retry the conversion. For more information, see Choosing optimization strategies and rules for use with AWS SCT (p. 158).

Related topics

- Choose the best sort key
- Choose the best distribution style

Creating and using the assessment report in AWS SCT

The AWS Schema Conversion Tool 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.
Database migration assessment report

Source database: GOLD_TEST_SS_PGTU-21-76.eu-west-1.compute.amazonaws.com\GOLD_TEST_SS_PG:1433
Microsoft SQL Server 2019 (RTM-CU10) (KB5001090) - 15.0.4123.1 (X64) Mar 22 2021 18:10:24
Copyright (C) 2019 Microsoft Corporation
Enterprise Edition: Core-based Licensing (64-bit) on Windows Server 2019 Datacenter 10.0 <X64> (Build 17763:) (Hypervisor)
Case sensitivity: OFF

Executive summary

We completed the analysis of your Microsoft SQL Server source database and estimate that 90% of the database storage objects and 77% of database code objects require 100 complex user action(s) to complete the conversion. Based on the source code syntax analysis, we estimate 94% (based on # lines of code) of your code can be converted to Amazon RDS for SQL Server.

Migration guidance for database objects that could not be converted automatically can be found here

Database objects with conversion actions for Amazon RDS for SQL Server

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identified 529 (90%) database storage objects and 1,215 (80%) database code objects that require 100 complex user action(s) to complete the conversion.

We found 7 encrypted object(s).

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

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

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple actions is likely to require more complex actions if any of them are complex.

Figure: Conversion statistics for database storage objects
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 AWS SCT** (p. 176).
### Action items

1. **Issue: 609: MySQL doesn't support the OUTPUT parameter.**
   - Recommended action: Create a trigger for INSERT statements for @Output PosNo.
   - Number of occurrences: 1
   - Documentation reference(s): [URL](https://example.com)

2. **Issue: 681: MySQL doesn't support creating indexes on non-clustered columns.**
   - Recommended action: Use non-clustered indexes.
   - Number of occurrences: 2

3. **Issue: 794: MySQL doesn't support user-defined data types.**
   - Recommended action: Please review generated code and modify as necessary.
   - Number of occurrences: 1
   - Parameter: @InputPosNo (Number of occurrences: 1)

MySQL doesn't support user-defined data types. The user defined data type is used in the following objects:

- **Issue: 826: Check the default value for a DateTime.**
  - Recommended action: Check the default value for a `DateTime` variable.
  - Number of occurrences: 1

- **Issue: 844: MySQL expands fractional seconds support.**
  - Recommended action: Review your transformed code and modifications.
  - Number of occurrences: 8
  - Documentation reference(s): [URL](https://example.com)

- **Issue: 9997: Unable to resolve objects.**
  - Recommended action: Verify if the unresolved object is present in your project.
  - Number of occurrences: 3

- **Issue: 690: MySQL doesn't support table types.**
  - Recommended action: Perform a manual conversion.
  - Number of occurrences: 1

- **Issue: 811: Unable to convert functions.**
  - Recommended action: Create a user-defined function.
  - Number of occurrences: 12

---

**Source Microsoft SQL Server procedure:** `POSITION_UPDATE_CASH_CGT_BULK`  

```sql
CREATE PROCEDURE POSITION_UPDATE_CASH_CGT_BULK
  @InputPosNo tvpPosNo readonly,
  @posFlags bigint = 0,
  @posFlagMask bigint = 0
AS
UPDATE p
SET p.Flags = p.Flags & (~@posFlagMask)
FROM Position p
  INNER JOIN @InputPosNo ipn ON p.ipn = ipn.ipn
RETURN 0
```
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.
Database objects with conversion actions for Amazon RDS PostgreSQL

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

We found 7 encrypted object(s).

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

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

The object actions complexity is a sum of the complexity of the action items associated with the object.

Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

![Conversion statistics for database storage objects](image)
Handling manual conversions in AWS SCT

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 AWS SCT (p. 177).

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 AWS SCT

You can update both the source schema and the target schema in your AWS Schema Conversion Tool 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**.

## Saving and applying your converted schema in AWS SCT

When the AWS Schema Conversion Tool generates converted schema (as shown in Converting your schema using AWS SCT (p. 163)), 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 migration assessment reports with AWS SCT (p. 125).

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 DB instance to apply your converted schema to your target database.

#### To save your converted schema as SQL scripts

1. Choose your schema and open the context (right-click) menu.
2. Choose **Save as SQL**.
3. Enter the name of the file and choose **Save**.
4. Save your converted schema using one of the following options:
   - Single file
   - Single file per stage
   - Single file per statement

#### To choose the format of the SQL script

1. On the **Settings** menu, choose **Project settings**.
2. Choose **Save scripts**.
3. For **Vendor**, choose the database platform.
4. For **Save SQL scripts to**, choose how you want to save your database schema script.
5. Choose **OK** to save the settings.
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`
- Snowflake: `aws_snowflake_ext`
- Teradata: `aws_teradata_ext`
- Vertica: `aws_vertica_ext`

For more information, see **Using AWS SCT extension packs (p. 296)**.

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 AWS SCT extension packs (p. 296)**.
Optimizing Amazon Redshift by using AWS SCT

You can use the AWS Schema Conversion Tool 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.
Converting extract, transform, and load (ETL) processes with AWS Schema Conversion Tool

You can use the AWS Schema Conversion Tool (AWS SCT) to migrate extract, transform, and load (ETL) processes. This type of migration includes the conversion of ETL-related business logic. This logic can reside either inside your source data warehouses or in external scripts that you run separately.

Currently, AWS SCT supports the conversion of ETL scripts to objects to AWS Glue and Amazon Redshift RSQL, as shown in the following table.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server Integration Services (SSIS) ETL packages</td>
<td>AWS Glue or AWS Glue Studio</td>
</tr>
<tr>
<td>Microsoft SQL Server stored database code objects</td>
<td>AWS Glue</td>
</tr>
<tr>
<td>Oracle stored database code objects</td>
<td>AWS Glue</td>
</tr>
<tr>
<td>Shell scripts with embedded commands from Teradata Basic Teradata Query (BTEQ)</td>
<td>Amazon Redshift RSQL</td>
</tr>
<tr>
<td>Teradata BTEQ ETL scripts</td>
<td>AWS Glue or Amazon Redshift RSQL</td>
</tr>
<tr>
<td>Teradata FastExport job scripts</td>
<td>Amazon Redshift RSQL</td>
</tr>
<tr>
<td>Teradata FastLoad job scripts</td>
<td>Amazon Redshift RSQL</td>
</tr>
<tr>
<td>Teradata MultiLoad job scripts</td>
<td>Amazon Redshift RSQL</td>
</tr>
<tr>
<td>Teradata stored database code objects</td>
<td>AWS Glue</td>
</tr>
</tbody>
</table>

Topics

- Converting ETL processes to AWS Glue with AWS SCT (p. 182)
- Converting ETL processes using the Python API for AWS Glue with AWS SCT (p. 186)
- Converting SSIS to AWS Glue with AWS SCT (p. 189)
- Converting SSIS to AWS Glue Studio with AWS SCT (p. 191)
- Converting Teradata BTEQ scripts to Amazon Redshift RSQL with AWS SCT (p. 194)
- Converting shell scripts with embedded Teradata BTEQ commands to Amazon Redshift RSQL with AWS SCT (p. 198)
- Converting Teradata FastExport job scripts to Amazon Redshift RSQL with AWS SCT (p. 202)
- Converting Teradata FastLoad job scripts to Amazon Redshift RSQL with AWS SCT (p. 205)
Converting ETL processes to AWS Glue with AWS SCT

Following, you can find an outline of the process to convert ETL scripts to AWS Glue with AWS SCT. For this example, we convert an Oracle database to Amazon Redshift, along with the ETL processes used with the source databases and data warehouses.

Topics
- Prerequisites (p. 183)
- Understanding the AWS Glue Data Catalog (p. 183)
- Limitations for converting using AWS SCT with AWS Glue (p. 183)
- Step 1: Create a new project (p. 184)
- Step 2: Create an AWS Glue job (p. 185)

The following architecture diagram shows an example database migration project that includes the conversion of ETL scripts to AWS Glue.
Prerequisites

Before you begin, do the following:

- Migrate any source databases that you intend to migrate to AWS.
- Migrate the target data warehouses to AWS.
- Collect a list of all the code involved in your ETL process.
- Collect a list of all the necessary connection information for each database.

Understanding the AWS Glue Data Catalog

As part of the process of conversion, AWS Glue loads information regarding the source and target databases. It organizes this information into categories, in a structure called a tree. The structure includes the following:

- **Connections** – Connection parameters
- **Crawlers** – A list of crawlers, one crawler for each schema
- **Databases** – Containers that hold tables
- **Tables** – Metadata definitions that represent the data in the tables
- **ETL jobs** – Business logic that performs the ETL work
- **Triggers** – Logic that controls when an ETL job runs in AWS Glue (whether on demand, by schedule, or triggered by job events)

The **AWS Glue Data Catalog** is an index to the location, schema, and runtime metrics of your data. When you work with AWS Glue and AWS SCT, the AWS Glue Data Catalog contains references to data that is used as sources and targets of your ETL jobs in AWS Glue. To create your data warehouse, catalog this data.

You use the information in the Data Catalog to create and monitor your ETL jobs. Typically, you run a crawler to take inventory of the data in your data stores, but there are other ways to add metadata tables into your Data Catalog.

When you define a table in your Data Catalog, you add it to a database. A database is used to organize tables in AWS Glue.

Limitations for converting using AWS SCT with AWS Glue

The following limitations apply when converting using AWS SCT with AWS Glue.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of databases for each account</td>
<td>10,000</td>
</tr>
<tr>
<td>Number of tables for each database</td>
<td>100,000</td>
</tr>
<tr>
<td>Number of partitions for each table</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Number of table versions for each table</td>
<td>100,000</td>
</tr>
</tbody>
</table>
### Step 1: Create a new project

To create a new project, take these high-level steps:

1. Create a new project in AWS SCT. For more information, see Creating an AWS SCT project (p. 16).
2. Add your source and target databases to the project. For more information, see Adding database servers to an AWS SCT project (p. 17).

Make sure that you have chosen **Use AWS Glue** in the target database connection settings. To do so, choose the **AWS Glue** tab. For **Copy from AWS profile**, choose the profile that you want to use. The profile should automatically fill in the AWS access key, secret key, and Amazon S3 bucket folder. If it doesn’t, enter this information yourself. After you choose **OK**, AWS Glue analyzes the objects and loads metadata into the AWS Glue Data Catalog.

Depending on your security settings, you might get a warning message that says your account doesn’t have sufficient privileges for some of the schemas on the server. If you have access to the schemas that you’re using, you can safely ignore this message.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tables for each account</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Number of partitions for each account</td>
<td>10,000,000</td>
</tr>
<tr>
<td>Number of table versions for each account</td>
<td>1,000,000</td>
</tr>
<tr>
<td>Number of connections for each account</td>
<td>1,000</td>
</tr>
<tr>
<td>Number of crawlers for each account</td>
<td>25</td>
</tr>
<tr>
<td>Number of jobs for each account</td>
<td>25</td>
</tr>
<tr>
<td>Number of triggers for each account</td>
<td>25</td>
</tr>
<tr>
<td>Number of concurrent job runs for each account</td>
<td>30</td>
</tr>
<tr>
<td>Number of concurrent job runs for each job</td>
<td>3</td>
</tr>
<tr>
<td>Number of jobs for each trigger</td>
<td>10</td>
</tr>
<tr>
<td>Number of development endpoints for each account</td>
<td>5</td>
</tr>
<tr>
<td>Maximum data processing units (DPUs) used by a development endpoint at one time</td>
<td>5</td>
</tr>
<tr>
<td>Maximum DPUs used by a role at one time</td>
<td>100</td>
</tr>
<tr>
<td>Database name length</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Database name length</td>
<td></td>
</tr>
<tr>
<td>For compatibility with other metadata stores, such as Apache Hive, the name is changed to use lowercase characters.</td>
<td></td>
</tr>
<tr>
<td>If you plan to access the database from Amazon Athena, provide a name with only alphanumeric and underscore characters.</td>
<td></td>
</tr>
<tr>
<td>Connection name length</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Crawler name length</td>
<td>Unlimited</td>
</tr>
</tbody>
</table>
3. To finish preparing to import your ETL, connect to your source and target databases. To do so, choose your database in the source or target metadata tree, and then choose **Connect to the server**.

AWS Glue creates a database on the source database server and one on the target database server to help with the ETL conversion. The database on the target server contains the AWS Glue Data Catalog. To find specific objects, use search on the source or target panels.

To see how a specific object converts, find an item you want to convert, and choose **Convert schema** from its context (right-click) menu. AWS SCT transforms this selected object into a script.

You can review the converted script from the **Scripts** folder in the right panel. Currently, the script is a virtual object, which is available only as part of your AWS SCT project.

To create an AWS Glue job with your converted script, upload your script to Amazon S3. To upload the script to Amazon S3, choose the script, then choose **Save to S3** from its context (right-click) menu.

**Step 2: Create an AWS Glue job**

After you save the script to Amazon S3, you can choose it and then choose **Configure AWS Glue Job** to open the wizard to configure the AWS Glue job. The wizard makes it easier to set this up:

1. On the first tab of the wizard, **Design Data Flow**, you can choose an execution strategy and the list of scripts you want to include in this one job. You can choose parameters for each script. You can also rearrange the scripts so that they run in the correct order.

2. On the second tab, you can name your job, and directly configure settings for AWS Glue. On this screen, you can configure the following settings:
   - AWS Identity and Access Management (IAM) role
   - Script file names and file paths
   - Encrypt the script using server-side encryption with Amazon S3–managed keys (SSE-S3)
   - Temporary directory
   - Generated Python library path
   - User Python library path
   - Path for the dependent .jar files
   - Referenced files path
   - Concurrent DPUs for each job run
   - Maximum concurrency
   - Job timeout (in minutes)
   - Delay notification threshold (in minutes)
   - Number of retries
   - Security configuration
   - Server-side encryption

3. On the third step, or tab, you choose the configured connection to the target endpoint.

After you finish configuring the job, it displays under the ETL jobs in the AWS Glue Data Catalog. If you choose the job, the settings display so you can review or edit them. To create a new job in AWS Glue, choose **Create AWS Glue Job** from the context (right-click) menu for the job. Doing this applies the schema definition. To refresh the display, choose **Refresh from database** from the context (right-click) menu.

At this point, you can view your job in the AWS Glue console. To do so, sign in to the AWS Management Console and open the AWS Glue console at [https://console.aws.amazon.com/glue/](https://console.aws.amazon.com/glue/).
Converting ETL processes using the Python API for AWS Glue with AWS SCT

In the following sections, you can find a description of a conversion that calls AWS Glue API operations in Python. For more information, see Program AWS Glue ETL scripts in Python in the AWS Glue Developer Guide.

Step 1: Create a database

The first step is to create a new database in an AWS Glue Data Catalog by using the AWS SDK API. When you define a table in the Data Catalog, you add it to a database. A database is used to organize the tables in AWS Glue.

The following example demonstrates the `create_database` method of the Python API for AWS Glue.

```python
response = client.create_database(
    DatabaseInput={
        'Name': 'database_name',
        'Description': 'description',
        'LocationUri': 'string',
        'Parameters': {
            'parameter-name': 'parameter value'
        }
    }
)
```

If you are using Amazon Redshift, the database name is formed as follows.

```
{redshift_cluster_name}_{redshift_database_name}_{redshift_schema_name}
```

The full name of Amazon Redshift cluster for this example is as follows.

```
rsdbb03.apqlmpqso.us-west-2.redshift.amazonaws.com
```

The following shows an example of a well-formed database name. In this case `rsdbb03` is the name, which is the first part of the full name of the cluster endpoint. The database is named `dev` and the schema is `ora_glue`.

```
rsdbb03_dev_ora_glue
```

Step 2: Create a connection

Create a new connection in a Data Catalog by using the AWS SDK API.

The following example demonstrates using the `create_connection` method of the Python API for AWS Glue.
response = client.create_connection(
    ConnectionInput={
        'Name': 'Redshift_abcede03.aabbcc112233.us-west-2.redshift.amazonaws.com_dev',
        'Description': 'Created from SCT',
        'ConnectionType': 'JDBC',
        'ConnectionProperties': {
            'JDBC_CONNECTION_URL': 'jdbc:redshift://aabbcc03.aabbcc112233.us-west-2.redshift.amazonaws.com:5439/dev',
            'USERNAME': 'user_name',
            'PASSWORD': 'password'
        },
        'PhysicalConnectionRequirements': {
            'AvailabilityZone': 'us-west-2c',
            'SubnetId': 'subnet-a1b23c45',
            'SecurityGroupIdList': ['sg-000a2b3c', 'sg-1a230b4c', 'sg-aba12c3d', 'sg-1abb2345']
        }
    }
)

The parameters used in `create_connection` are as follows:

- **Name** (UTF-8 string) – required. For Amazon Redshift, the connection name is formed as follows: `Redshift_{Endpoint-name}_{redshift-database-name}`, for example: `Redshift_abcede03_dev`
- **Description** (UTF-8 string) – Your description of the connection.
- **ConnectionType** (UTF-8 string) – Required. The type of connection. Currently, only JDBC is supported; SFTP is not supported.
- **ConnectionProperties** (dict) – Required. A list of key-value pairs used as parameters for this connection, including the JDBC connection URL, the user name, and the password.
- **PhysicalConnectionRequirements** (dict) – Physical connection requirements, which include the following:
  - **SubnetId** (UTF-8 string) – The ID of the subnet used by the connection.
  - **SecurityGroupIdList** (list) – The security group ID list used by the connection.
  - **AvailabilityZone** (UTF-8 string) – Required. The Availability Zone that contains the endpoint. This parameter is deprecated.

### Step 3: Create an AWS Glue crawler

Next, you create an AWS Glue crawler to populate the AWS Glue catalog. For more information, see [Cataloging tables with a crawler](https://docs.aws.amazon.com/glue/latest/dg/aws-glue-crawling.html) in the *AWS Glue Developer Guide*.

The first step in adding a crawler is to create a new database in a Data Catalog by using the AWS SDK API. Before you begin, make sure to first delete any previous version of it by using the `delete_crawler` operation.

When you create your crawler, a few considerations apply:

- For the crawler name, use the format `<redshift_node_name>_<redshift_database_name>_<redshift_shema_name>`, for example: `abcede03_dev_ora_glue`
- Use an IAM role that already exists. For more information on creating IAM roles, see [Creating IAM roles](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html) in the *IAM User Guide*.
- Use the name of the database that you created in the previous steps.
Step 3: Create an AWS Glue crawler

- Use the `ConnectionName` parameter, which is required.
- For the `path` parameter, use the path to the JDBC target, for example: `dev/ora_glue/%`

The following example deletes an existing crawler and then creates a new one by using the Python API for AWS Glue.

```python
response = client.delete_crawler(
    Name='crawler_name'
)
response = client.create_crawler(
    Name='crawler_name',
    Role= 'IAM_role',
    DatabaseName='database_name',
    Description='string',
    Targets={
        'S3Targets': [
            {
                'Path': 'string',
                'Exclusions': ['string'],
            }
        ],
        'JdbcTargets': [
            {
                'ConnectionName': 'ConnectionName',
                'Path': 'Include_path',
                'Exclusions': ['string'],
            }
        ],
    },
    Schedule='string',
    Classifiers=[
       'string',
    ],
    TablePrefix='string',
    SchemaChangePolicy={
        'UpdateBehavior': 'LOG'|'UPDATE_IN_DATABASE',
        'DeleteBehavior': 'LOG'|'DELETE_FROM_DATABASE'|'DEPRECATE_IN_DATABASE'
    },
    Configuration='string'
)
```

Create and then run a crawler that connects to one or more data stores, determines the data structures, and writes tables into the Data Catalog. You can run your crawler on a schedule, as shown following.

```python
response = client.start_crawler(
    Name='string'
)
```

This example uses Amazon Redshift as the target. Amazon Redshift data types map to AWS Glue data types in the following way after the crawler runs.

<table>
<thead>
<tr>
<th>Amazon Redshift data type</th>
<th>AWS Glue data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallint</td>
<td>smallint</td>
</tr>
</tbody>
</table>
Converting SSIS to AWS Glue with AWS SCT

Following, you can find how to convert Microsoft SQL Server Integration Services (SSIS) packages to AWS Glue using AWS SCT.

To convert Microsoft SSIS packages to AWS Glue, make sure that you use AWS SCT version 1.0.642 or later. You also need to have an SSIS project with ETL packages – .dtsx, .conmgr, and .params files in the local folder.

You don’t need an installed SSIS server. The conversion process goes through the local SSIS files.

**To convert an SSIS package to AWS Glue using AWS SCT**

1. Create a new project in AWS SCT or open an existing project. For more information, see the section called “Creating a project” (p. 16).
2. Choose **Add source** from the menu to add a new source SSIS package to your project.
3. Choose **SQL Server Integration Services** and complete the following:
   - **Connection name** – Enter the name for your connection. AWS SCT displays this name in the metadata tree.
   - **SSIS packages folder** – Choose the path to your SSIS project folder with packages.

   AWS SCT reads the project files (files with the extensions .dtsx, .conmgr or .params) from the local folder and parses them. It then organizes them into an AWS SCT tree of categories.

4. Choose **Add target** from the menu to add a new target platform to convert your source SSIS packages.
5. Choose **AWS Glue** and complete the following:
   - **Connection name** – Enter the name for your connection. AWS SCT displays this name in the metadata tree.

### Data Type Conversion

<table>
<thead>
<tr>
<th>SSIS Type</th>
<th>AWS Glue Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>int</td>
</tr>
<tr>
<td>bigint</td>
<td>bigint</td>
</tr>
<tr>
<td>decimal</td>
<td>decimal(18,0)</td>
</tr>
<tr>
<td>decimal(p,s)</td>
<td>decimal(p,s)</td>
</tr>
<tr>
<td>real</td>
<td>double</td>
</tr>
<tr>
<td>double precision</td>
<td>double</td>
</tr>
<tr>
<td>boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>char</td>
<td>string</td>
</tr>
<tr>
<td>varchar</td>
<td>string</td>
</tr>
<tr>
<td>varchar(n)</td>
<td>string</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
</tr>
<tr>
<td>timestamp</td>
<td>timestamp</td>
</tr>
<tr>
<td>timestamptz</td>
<td>timestamp</td>
</tr>
</tbody>
</table>
• **Copy from AWS profile** – Choose the profile to use.
• **AWS access key** – Enter your AWS access key.
• **AWS secret key** – Enter your AWS secret key.
• **Region** – Choose the AWS Region that you want to use from the list.
• **Amazon S3 bucket folder** – Enter the folder path for the Amazon S3 bucket that you plan to use.

You can use a virtual AWS Glue target. In this case, you don’t need to specify the connection credentials. For more information, see the section called “Virtual targets” (p. 123).

6. Create a new mapping rule that includes your source SSIS package and your AWS Glue target. For more information, see the section called “New rule” (p. 122).

7. On the **View** menu, choose **Main view**.

8. In the SSIS tree view, open the context (right-click) menu for **Connection managers**, and then choose **Configure connections**.

9. Configure the project connection manager.

   To configure a connection mapping for SSIS connection managers, specify the AWS Glue connection for the corresponding SSIS connection manager. Make sure that your AWS Glue connections are already created.

   a. Under **Connections**, choose **Project connections**.
   b. For **Glue catalog connection**, choose the appropriate AWS Glue connection.

10. Configure the package connection manager:

    a. Under **Connections**, choose your package.
    b. For **Glue catalog connection**, choose the appropriate AWS Glue connection.
    c. Repeat these actions for all connections available for your package.

11. Choose **Apply**.

12. Convert your package. In the source tree view, find **Packages**. Open the context (right-click) menu for your package, then choose **Convert package**.

13. Save the converted script to Amazon S3. In the target tree view, find **Package scripts**. Open the context (right-click) menu for your converted script, then choose **Save to S3**.

14. Configure your AWS Glue job. In the target tree view, find **Package scripts**. Open the context (right-click) menu for your converted script, then choose **Configure AWS Glue job**.

15. Complete the three configuration sections:

    a. Complete the **Design data flow** section:

        • **Execution strategy** – Choose how your job will run ETL scripts. Choose **SEQUENTIAL** to run the scripts in the order that is specified in the wizard. Choose **PARALLEL** to run the scripts in parallel, disregarding the order that is specified in the wizard.
        • **Scripts** – Choose the name of your converted script.
        • Choose **Next**.

    b. Complete the **Job properties** section:

        • **Name** – Enter the name of your AWS Glue job.
        • **IAM Role** – Choose the IAM role that is used for authorization to resources used to run the job and access data stores.
        • **Script file name** – Enter the name of your converted script.
        • **Script file S3 path** – Enter the Amazon S3 path to your converted script.
        • **Encrypt script using SSE-S3** – Choose this option to protect data using server-side encryption with Amazon S3-managed encryption keys (SSE-S3).
- **Temporary directory** – Enter the Amazon S3 path to a temporary directory for intermediate results. AWS Glue and AWS Glue built-in transforms use this directory to read or write to Amazon Redshift.
- AWS SCT automatically generates the path for Python libraries. You can review this path in *Generated python library path*. You can’t edit this automatically generated path. To use additional Python libraries, enter the path in *User python library path*.
- **User python library path** – Enter the paths for additional user Python libraries. Separate Amazon S3 paths with commas.
- **Dependent jars path** – Enter the paths for dependent jar files. Separate Amazon S3 paths with commas.
- **Referenced files path** – Enter the paths for additional files, such as configuration files, that are required by your script. Separate Amazon S3 paths with commas.
- **Maximum capacity** – Enter the maximum number of AWS Glue data processing units (DPUs) that can be allocated when this job runs. You can enter an integer from 2 to 100. The default is 2.
- **Max concurrency** – Enter the maximum number of concurrent runs that are allowed for this job. The default is 1. AWS Glue returns an error when this threshold is reached.
- **Job timeout (minutes)** – Enter the timeout value on your ETL job as a safeguard against runaway jobs. The default is 2880 minutes (48 hours) for batch jobs. If the job exceeds this limit, the job run state changes to **TIMEOUT**.
- **Delay notification threshold (minutes)** – Enter the threshold in minutes before AWS SCT sends a delay notification.
- **Number of retries** – Enter the number of times (0–10) that AWS Glue should automatically restart the job if it fails. Jobs that reach the timeout limit aren’t restarted. The default is 0.

Choose **Next**.

c. Configure the required connections:

i. From **All connections**, choose the required AWS Glue connections and add them to the list of **Selected connections**.

ii. Choose **Finish**.

16. Create a configured AWS Glue job. In the target tree view, find and expand **ETL Jobs**. Open the context (right-click) menu for ETL job that you configured, and then choose **Create AWS Glue Job**.

17. Run the AWS Glue job:


b. In the navigation pane, choose **Jobs**.

c. Choose **Add job**, and then choose the job that you want to run.

d. On the **Actions** tab, choose **Run job**.

### Converting SSIS to AWS Glue Studio with AWS SCT

You can use AWS SCT to convert Microsoft SQL Server Integration Services (SSIS) packages to AWS Glue Studio.

An **SSIS package** includes the necessary components, such as the connection manager, tasks, control flow, data flow, parameters, event handlers, and variables, to run a specific extract, transform, and load (ETL) task. AWS SCT converts SSIS packages to a format compatible with AWS Glue Studio. After you migrate your source database to the AWS Cloud, you can run these converted AWS Glue Studio jobs to perform ETL tasks against the new database.

To convert Microsoft SSIS packages to AWS Glue Studio, make sure that you use AWS SCT version 1.0.661 or later.
Adding SSIS packages to your AWS SCT project

You can add multiple SSIS packages to a single AWS SCT project.

To add an SSIS package to your AWS SCT project

1. Create a new project in AWS SCT or open an existing project. For more information, see the section called “Creating a project” (p. 16).
2. Choose Add source from the menu, and then choose SQL Server Integration Services.
3. For Connection name, enter a name for your SSIS package. AWS SCT displays this name in the tree in the left panel.
4. For SSIS packages folder, enter the path to the folder with source SSIS packages.
5. Choose Add target from the menu, and then choose AWS Glue Studio.
   To connect to AWS Glue Studio, AWS SCT uses your AWS profile. For more information, see Storing AWS service profiles in the AWS SCT (p. 34).
6. Create a mapping rule, which includes your source SSIS package and your AWS Glue Studio target. For more information, see Creating mapping rules in AWS SCT (p. 122).
7. Create AWS Glue Studio connections in the AWS Glue Studio console. For more information, see Creating connections for connectors.
8. Choose Connection managers in the left tree, open the context (right-click) menu, and then choose Configure connections.
   AWS SCT displays the Configure connections window.
9. For each source SSIS connection, choose an AWS Glue Studio connection.

Converting SSIS packages to AWS Glue Studio with AWS SCT

Following, find how to convert SSIS packages to AWS Glue Studio using AWS SCT.

To convert an SSIS package to AWS Glue Studio

1. Add your SSIS package to your AWS SCT project. For more information, see Adding SSIS packages to your AWS SCT project (p. 192).
2. In the left panel, expand the ETL and SSIS nodes.
3. Choose Packages, open the context (right-click) menu, and then choose Convert package.
   AWS SCT converts your selected SSIS packages to JSON files. These JSON objects represent a node in a directed acyclic graph (DAG). Find your converted files in the Package DAGs node in the right tree.
4. Choose Package DAGs, open the context (right-click) menu, and then choose Save to Amazon S3.
   Now you can use these scripts to create jobs in the AWS Glue Studio.

Creating AWS Glue Studio jobs using the converted code

After you convert your source SSIS packages, you can use the converted JSON files to create AWS Glue Studio jobs.
To create an AWS Glue Studio job

1. Choose Package DAGs in the right tree, open the context (right-click) menu, and then choose Configure AWS Glue Studio job.

2. (Optional) Apply the extension pack that emulates SSIS functions in AWS Glue Studio.

3. The Configure AWS Glue Studio job window opens.

Complete the Basic job properties section:

• **Name** – Enter a name of your AWS Glue Studio job.
• **Script file name** – Enter a name of your job script.
• **Job parameters** – Add parameters and enter their values.

Choose Next.

4. Complete the Advanced job properties section:

• **IAM Role** – Choose the IAM role that is used for authorization to AWS Glue Studio and access data stores.
• **Script file S3 path** – Enter the Amazon S3 path to your converted script.
• **Temporary directory** – Enter the Amazon S3 path to a temporary directory for intermediate results. AWS Glue Studio uses this directory to read or write to Amazon Redshift.
• **AWS SCT automatically generates the path for Python libraries. You can review this path in Generated python library path. You can't edit this automatically generated path. To use additional Python libraries, enter the path in User python library path.
• **User python library path** – Enter the paths for additional user Python libraries. Separate Amazon S3 paths with commas.
• **Dependent jars path** – Enter the paths for dependent *.jar files. Separate Amazon S3 paths with commas.
• **Referenced files path** – Enter the paths for additional files, such as configuration files, that are required by your script. Separate Amazon S3 paths with commas.
• **Worker type** – Choose G.1X or G.2X.

When you choose G.1X each worker maps to 1 DPU (4 vCPU, 16 GB of memory, and 64 GB disk).

When you choose G.2X each worker maps to 2 DPU (8 vCPU, 32 GB of memory, and 128 GB disk).
• **Requested number of workers** – Enter the number of workers that are allocated when the job runs.
• **Max concurrency** – Enter the maximum number of concurrent runs that are allowed for this job. The default is 1. AWS Glue returns an error when this threshold is reached.
• **Job timeout (minutes)** – Enter the timeout value on your ETL job as a safeguard against runaway jobs. The default is 2,880 minutes (48 hours) for batch jobs. If the job exceeds this limit, the job run state changes to TIMEOUT.
• **Delay notification threshold (minutes)** – Enter the threshold in minutes before AWS SCT sends a delay notification.
• **Number of retries** – Enter the number of times (0–10) that AWS Glue should automatically restart the job if it fails. Jobs that reach the timeout limit aren't restarted. The default is 0.

Choose Finish.

AWS SCT configures your selected AWS Glue Studio jobs.

5. Find your configured jobs under ETL jobs in the right tree. Choose your configured job, open the context (right-click) menu, and then choose Create AWS Glue Studio job.
6. Choose **Apply status** and make sure that the **Status** value for your job is **Success**.
7. Open the AWS Glue Studio console, choose **Refresh**, and choose your job. Then choose **Run**.

Creating an assessment report for an SSIS package with AWS SCT

The **ETL migration assessment report** provides information about converting your SSIS packages to a format compatible with AWS Glue Studio. The assessment report includes action items for the components of your SSIS packages. These action items show which components AWS SCT can't automatically convert.

**To create an ETL migration assessment report**

1. Expand the **SSIS** node under **ETL** in the left panel.
2. Choose **Packages**, open the context (right-click) menu, and then choose **Create report**.
3. View the **Summary** tab. Here, AWS SCT displays the executive summary information from the ETL migration assessment report. It includes conversion results for all components of your SSIS packages.
4. (Optional) Save a local copy of the ETL migration assessment report as either a PDF file or a comma-separated values (CSV) file:
   - To save the ETL migration assessment report as a PDF file, choose **Save to PDF** at upper right. The PDF file contains the executive summary, action items, and recommendations for scripts conversion.
   - To save the ETL migration assessment report as a CSV file, choose **Save to CSV** at upper right. AWS SCT creates three CSV files. These files contain action items, recommended actions, and an estimated complexity of manual effort required to convert the scripts.
5. Choose the **Action items** tab. This tab contains a list of items that require manual conversion to AWS Glue Studio. When you choose an action item from the list, AWS SCT highlights the item from your source SSIS package that the action item applies to.

Converting Teradata BTEQ scripts to Amazon Redshift RSQL with AWS SCT

You can use the AWS Schema Conversion Tool (AWS SCT) to convert Teradata Basic Teradata Query (BTEQ) scripts to Amazon Redshift RSQL.

The following architecture diagram shows the database migration project that includes the conversion of extract, transform, and load (ETL) scripts to Amazon Redshift RSQL.
Adding BTEQ scripts to your AWS SCT project

You can add multiple scripts to a single AWS SCT project.

To add a BTEQ script to your AWS SCT project

1. Create a new project in AWS SCT or open an existing project. For more information, see the section called “Creating a project” (p. 16).
2. Choose Add source from the menu, and then choose Teradata to add your source database to the project. For more information, see Using Teradata as a source (p. 117).
3. Choose Add target from the menu to add a target Amazon Redshift database to your AWS SCT project.
   
   You can use a virtual Amazon Redshift target database platform. For more information, see Using virtual targets (p. 123).
4. Create a new mapping rule that includes your source Teradata database and your Amazon Redshift target. For more information, see Adding a new mapping rule (p. 122).
5. On the View menu, choose Main view.
6. In the left panel, expand the Scripts node.
7. Choose BTEQ scripts, open the context (right-click) menu, and then choose Load scripts.
8. Enter the location of the source code for your Teradata BTEQ scripts and choose **Select folder**. AWS SCT displays the **Load scripts** window.

9. Do one of the following:
   a. If your Teradata BTEQ scripts don't include the substitution variables, choose **No substitution variables**, and then choose **OK** to add scripts to your AWS SCT project.
   b. If your Teradata BTEQ scripts include the substitution variables, configure the substitution variables. For more information, see **Configuring substitution variables in BTEQ scripts** (p. 196).

## Configuring substitution variables in BTEQ scripts with AWS SCT

Your Teradata BTEQ scripts can include substitution variables. For example, you can use one BTEQ script with substitution variables to run the same set of commands on multiple database environments. You can use AWS SCT to configure substitution variables in your BTEQ scripts.

Before you run a BTEQ script with substitution variables, make sure to assign the values for all variables. To do this, you can use other tools or applications such as a Bash script, UC4 (Automic), and so on. AWS SCT can resolve and convert substitution variables only after you assign their values.

**To configure substitution variables in your BTEQ script**

1. Add your BTEQ scripts to your AWS SCT project. For more information, see **Adding BTEQ scripts to your AWS SCT project** (p. 195).
   
   When you add your scripts, choose **Substitution variables are used**.

2. For **Define variable format**, enter a regular expression that matches all substitution variables in your script.
   
   For example, if the names of your substitution variables start with \${ and end with }, use the \$\{w+\} regular expression. To match substitution variables that start either with a dollar sign or a percent sign, use the $\w+|\%\w+ regular expression.

   Regular expressions in AWS SCT conform to the Java regular expression syntax. For more information, see **java.util.regex Class Pattern** in the Java documentation.

3. Choose **OK** to load scripts to your AWS SCT project, and then choose **OK** to close the **Load scripts** window.

4. Choose **Variables** to view all discovered substitution variables and their values.

5. For **Value**, enter the value for the substitution variable.

## Converting Teradata BTEQ scripts to Amazon Redshift RSQL with AWS SCT

Following, find how to convert BTEQ ETL scripts to Amazon Redshift RSQL using AWS SCT.

**To convert a Teradata BTEQ script to Amazon Redshift RSQL**

1. Add your BTEQ scripts to your AWS SCT project. For more information, see **Adding BTEQ scripts to your AWS SCT project** (p. 195).
2. Configure the substitution variables. For more information, see Configuring substitution variables in BTEQ scripts (p. 196).
3. In the left panel, expand the Scripts node.
4. Do one of the following:
   - To convert a single BTEQ script, expand the BTEQ scripts node, choose the script to convert, and then choose Convert to RSQL from the context (right-click) menu.
   - To covert multiple scripts, make sure that you select all scripts to convert. Then choose BTEQ scripts, open the context (right-click) menu, and then choose Convert to RSQL under Convert script.

AWS SCT converts all your selected Teradata BTEQ scripts to a format compatible with Amazon Redshift RSQL. Find your converted scripts in the Scripts node in the target database panel.
5. Edit your converted Amazon Redshift RSQL scripts, or save them. For more information, see Editing and saving your converted BTEQ scripts (p. 198).

Managing BTEQ scripts with AWS SCT

You can add multiple BTEQ scripts or remove a BTEQ script from your AWS SCT project.

To add an additional BTEQ script to your AWS SCT project
1. Expand the Scripts node in the left panel.
2. Choose the BTEQ scripts node, and open the context (right-click) menu.
3. Choose Load scripts.
4. Enter the information that is required to add a new BTEQ script and configure substitution variables. For more information, see Adding BTEQ scripts to your AWS SCT project (p. 195) and Configuring substitution variables in BTEQ scripts (p. 196).

To remove a BTEQ script from your AWS SCT project
1. Expand the BTEQ scripts node under Scripts in the left panel.
2. Choose the script to remove, and open the context (right-click) menu.
3. Choose Delete script.

Creating a BTEQ script conversion assessment report with AWS SCT

A BTEQ script conversion assessment report provides information about converting the BTEQ commands and SQL statements from your BTEQ scripts to a format compatible with Amazon Redshift RSQL. The assessment report includes action items for BTEQ commands and SQL statements that AWS SCT can't convert.

To create a BTEQ script conversion assessment report
1. Expand the BTEQ scripts node under Scripts in the left panel.
2. Choose the script to convert, and open the context (right-click) menu.
3. Choose Conversion to RSQL under Create report.
4. View the **Summary** tab. The **Summary** tab displays the executive summary information from the BTEQ script assessment report. It includes conversion results for all BTEQ commands and SQL statements from your BTEQ scripts.

5. (Optional) Save a local copy of the BTEQ script conversion assessment report as either a PDF file or a comma-separated values (CSV) file:
   - To save the BTEQ script conversion assessment report as a PDF file, choose **Save to PDF** at upper right.
     The PDF file contains the executive summary, action items, and recommendations for scripts conversion.
   - To save the BTEQ script conversion assessment report as a CSV file, choose **Save to CSV** at upper right.
     The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the scripts.

6. Choose the **Action items** tab. This tab contains a list of items that require manual conversion to Amazon Redshift RSQL. When you choose an action item from the list, AWS SCT highlights the item from your source BTEQ script that the action item applies to.

**Editing and saving your converted BTEQ scripts with AWS SCT**

You can edit your converted scripts in the lower panel of your AWS SCT project. AWS SCT stores the edited script as part of your project.

**To save your converted scripts**

1. Expand the **RSQL scripts** node under **Scripts** in the target database panel.
2. Choose your converted script, open the context (right-click) menu, and choose **Save script**.
3. Enter the path to the folder to save the converted script and choose **Save**.

AWS SCT saves the converted script to a file and opens this file.

**Converting shell scripts with embedded Teradata BTEQ commands to Amazon Redshift RSQL with AWS SCT**

You can use the AWS Schema Conversion Tool (AWS SCT) to convert shell scripts with embedded Teradata Basic Teradata Query (BTEQ) commands to shell scripts with embedded Amazon Redshift RSQL commands.

AWS SCT extracts Teradata BTEQ commands from your shell scripts and converts them to a format compatible with Amazon Redshift. After you migrate the Teradata database to Amazon Redshift, you can use these converted scripts to manage your new Amazon Redshift database.

You can also use AWS SCT to convert files with Teradata BTEQ ETL scripts to Amazon Redshift RSQL. For more information, see Converting Teradata BTEQ scripts to Amazon Redshift RSQL with AWS SCT (p. 194).
Adding shell scripts with embedded Teradata BTEQ commands to your AWS SCT project

You can add multiple scripts to a single AWS SCT project.

**To add a shell script to your AWS SCT project**

1. Create a new project in AWS SCT or open an existing project. For more information, see the section called “Creating a project” (p. 16).
2. Choose **Add source** from the menu, and then choose **Teradata** to add your source database to the project. For more information, see Using Teradata as a source (p. 117).
3. Choose **Add target** from the menu and to add a target Amazon Redshift database to your AWS SCT project.
   - You can use a virtual Amazon Redshift target database platform. For more information, see Using virtual targets (p. 123).
4. Create a new mapping rule that includes your source Teradata database and your Amazon Redshift target. For more information, see Adding a new mapping rule (p. 122).
5. On the **View** menu, choose **Main view**.
6. In the left panel, expand the **Scripts** node.
7. Choose **Shell**, open the context (right-click) menu, and then choose **Load scripts**.
8. Enter the location of your source shell scripts with embedded Teradata BTEQ commands and choose **Select folder**.
   - AWS SCT displays the **Load scripts** window.
9. Do one of the following:
   - If your shell scripts don't include the substitution variables, choose **No substitution variables**, and then choose **OK** to add scripts to your AWS SCT project.
   - If your shell scripts include the substitution variables, configure the substitution variables. For more information, see Configuring substitution variables in shell scripts (p. 199).

**Configuring substitution variables in shell scripts with embedded Teradata BTEQ commands with AWS SCT**

Your shell scripts can include substitution variables. For example, you can use a single script with substitution variables to manage databases in different environments. You can use AWS SCT to configure substitution variables in your shell scripts.

Before you run BTEQ commands with substitution variables from a shell script, make sure to assign the values for all variables inside this shell script. AWS SCT can resolve and convert substitution variables only after you assign their values.

**To configure substitution variables in your shell script**

1. Add your source shell scripts to your AWS SCT project. For more information, see Adding shell scripts to your AWS SCT project (p. 199).
   - When you add your scripts, choose **Substitution variables are used**.
2. For **Define variable format**, enter a regular expression that matches all substitution variables in your script.
For example, if the names of your substitution variables start with \$ and end with \}, use the \$\{\w+\} regular expression. To match substitution variables that start either with a dollar sign or a percent sign, use the \$\w+|\%\w+ regular expression.

Regular expressions in AWS SCT conform to the Java regular expression syntax. For more information, see java.util.regex Class Pattern in the Java documentation.

3. Choose OK to load scripts to your AWS SCT project, and then choose OK to close the Load scripts window.
4. Choose Variables to view all discovered substitution variables and their values.
5. For Value, enter the value for the substitution variable.

### Converting shell scripts with embedded Teradata BTEQ commands with AWS SCT

Following, find how to convert shell scripts with embedded Teradata BTEQ commands to shell scripts with embedded Amazon Redshift RSQL commands using AWS SCT.

#### To convert a shell script

1. Add your shell scripts to your AWS SCT project. For more information, see Adding shell scripts to your AWS SCT project (p. 199).
2. Configure the substitution variables. For more information, see Configuring substitution variables in shell scripts (p. 199).
3. In the left panel, expand the Scripts node.
4. Do one of the following:
   - To convert BTEQ commands from a single shell script, expand the Shell node, choose the script to convert, and then choose Convert script from the context (right-click) menu.
   - To covert multiple scripts, make sure that you select all scripts to convert. Then choose Shell, open the context (right-click) menu, and then choose Convert script.
5. Choose OK.

AWS SCT converts BTEQ commands in your selected shell scripts to a format compatible with Amazon Redshift RSQL. Find your converted scripts in the Scripts node in the target database panel.
6. Edit your converted Amazon Redshift RSQL scripts or save them. For more information, see Editing and saving your converted shell scripts (p. 201).

### Managing shell scripts with embedded Teradata BTEQ commands with AWS SCT

You can add multiple shell scripts or remove a shell script from your AWS SCT project.

#### To add a new shell script to your AWS SCT project

1. Expand the Scripts node in the left panel.
2. Choose the Shell node, and open the context (right-click) menu.
3. Choose Load scripts.
4. Enter the information that is required to add a new shell script and configure substitution variables. For more information, see Adding shell scripts to your AWS SCT project (p. 199) and Configuring substitution variables in shell scripts (p. 199).
To remove a shell script from your AWS SCT project

1. Expand the Shell node under Scripts in the left panel.
2. Choose the script to remove, and open the context (right-click) menu.
3. Choose Delete script.

Creating an assessment report for a shell script conversion with AWS SCT

The shell script conversion assessment report provides information about converting the BTEQ commands and SQL statements. The conversion is from your source scripts to a format compatible with Amazon Redshift RSQL. The assessment report includes action items for BTEQ commands and SQL statements that AWS SCT can’t convert.

To create a shell script conversion assessment report

1. Expand the Shell node under Scripts in the left panel.
2. Choose the script to convert, open the context (right-click) menu, and then choose Create report.
3. View the Summary tab. The Summary tab displays the executive summary information from the shell script assessment report. It includes conversion results for all BTEQ commands and SQL statements from your source scripts.
4. (Optional) Save a local copy of the shell script conversion assessment report as either a PDF file or a comma-separated values (CSV) file:
   - To save the shell script conversion assessment report as a PDF file, choose Save to PDF at upper right.
     The PDF file contains the executive summary, action items, and recommendations for scripts conversion.
   - To save the shell script conversion assessment report as a CSV file, choose Save to CSV at upper right.
     The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the scripts.
5. Choose the Action items tab. This tab contains a list of items that require manual conversion to Amazon Redshift RSQL. When you select an action item from the list, AWS SCT highlights the item from your source shell script that the action item applies to.

Editing and saving your converted shell scripts with AWS SCT

You can edit your converted scripts in the lower panel of your AWS SCT project. AWS SCT stores the edited script as part of your project.

To save your converted scripts

1. Expand the RSQL scripts node under Scripts in the target database panel.
2. Choose your converted script, open the context (right-click) menu, and choose Save script.
3. Enter the path to the folder to save the converted script and choose Save.

AWS SCT saves the converted script to a file and opens this file.
Converting Teradata FastExport job scripts to Amazon Redshift RSQL with AWS SCT

You can use the AWS Schema Conversion Tool (AWS SCT) to convert Teradata FastExport job scripts to Amazon Redshift RSQL.

A FastExport job script is a set of FastExport commands and SQL statements that select and export data from the Teradata database. AWS SCT converts FastExport commands and SQL statements to a format compatible with Amazon Redshift RSQL. After you migrate the Teradata database to Amazon Redshift, you can use these converted scripts to export data from the Amazon Redshift database.

Adding FastExport job scripts to your AWS SCT project

You can add multiple scripts to a single AWS SCT project.

To add a FastExport job script to your AWS SCT project

1. Create a new project in AWS SCT or open an existing project. For more information, see the section called “Creating a project” (p. 16).
2. Choose Add source from the menu, and then choose Teradata to add your source database to the project. For more information, see Using Teradata as a source (p. 117).
3. Choose Add target from the menu and to add a target Amazon Redshift database to your AWS SCT project.

You can use a virtual Amazon Redshift target database platform. For more information, see Using virtual targets (p. 123).
4. Create a new mapping rule that includes your source Teradata database and your Amazon Redshift target. For more information, see Adding a new mapping rule (p. 122).
5. On the View menu, choose Main view.
6. In the left panel, expand the Scripts node.
7. Choose FastExport, open the context (right-click) menu, and then choose Load scripts.
8. Enter the location of the source code for your Teradata FastExport job scripts and choose Select folder.

AWS SCT displays the Load scripts window.
9. Do one of the following:
   - If your Teradata FastExport job scripts don't include the substitution variables, choose No substitution variables and then choose OK to add scripts to your AWS SCT project.
   - If your Teradata FastExport job scripts include the substitution variables, configure the substitution variables. For more information, see Configuring substitution variables in FastExport job scripts (p. 202).

Configuring substitution variables in Teradata FastExport job scripts with AWS SCT

Your Teradata FastExport job scripts can include substitution variables. For example, you can use a single script with substitution variables to export data from multiple databases. You can use AWS SCT to configure substitution variables in your Teradata scripts.
Before you run a FastExport job script with substitution variables, make sure to assign the values for all variables. To do this, you can use other tools or applications such as a Bash script, UC4 (Automic), and so on. AWS SCT can resolve and convert substitution variables only after you assign their values.

To configure substitution variables in your FastExport job script

1. Add your source Teradata FastExport job scripts to your AWS SCT project. For more information, see Adding BTEQ scripts to your AWS SCT project (p. 195).

   When you add your scripts, choose Substitution variables are used.

2. For Define variable format, enter a regular expression that matches all substitution variables in your script.

   For example, if the names of your substitution variables start with `${` and end with `}`, use the `$\{\w+\}$` regular expression. To match substitution variables that start either with a dollar sign or a percent sign, use the `$\$\w+|\%\w+$` regular expression.

   Regular expressions in AWS SCT conform to the Java regular expression syntax. For more information, see java.util.regex Class Pattern in the Java documentation.

3. Choose OK to load scripts to your AWS SCT project, and then choose OK to close the Load scripts window.

4. In the left panel, expand the Scripts node. Choose FastExport, and then choose your folder with scripts. Open the context (right-click) menu, and then choose Export variables under Substitution variables.

5. Export substitution variables for one script. Expand your folder with scripts, choose your script, open the context (right-click) menu, and choose Export variables under Substitution variables.

6. Enter the name of the comma-separated values (CSV) file to save the substitution variables and choose Save.

7. Open this CSV file and fill in the values for the substitution variables.

   Depending on the operating system, AWS SCT uses different formats for CSV files. The values in the file might be either enclosed in quotation marks or not. Make sure that you use the same format for the values of substitution variables as the other values in the file. AWS SCT can’t import the CSV file with values in different formats.

8. Save the CSV file.

9. In the left panel, expand the Scripts node. Choose FastExport, and then choose your script. Open the context (right-click) menu, and then choose Import variables under Substitution variables.

10. Choose your CSV file, and then choose Open.

11. Choose Variables to view all discovered substitution variables and their values.

Converting Teradata FastExport job scripts with AWS SCT

Following, find how to convert Teradata FastExport job to Amazon Redshift RSQL using AWS SCT.

To convert a Teradata FastExport job script to Amazon Redshift RSQL

1. Add your FastExport job scripts to your AWS SCT project. For more information, see Adding FastExport job scripts to your AWS SCT project (p. 202).

2. Configure the substitution variables. For more information, see Configuring substitution variables in FastExport job scripts (p. 202).

3. In the left panel, expand the Scripts node.

4. Do one of the following:
• To convert a single FastExport job script, expand the **FastExport** node, choose the script to convert, and then choose **Convert script** from the context (right-click) menu.

• To convert multiple scripts, make sure that you select all scripts to convert. Then choose **FastExport**, open the context (right-click) menu, and then choose **Convert script**.

AWS SCT converts all your selected Teradata FastExport job scripts to a format compatible with Amazon Redshift RSQL. Find your converted scripts in the **Scripts** node in the target database panel.

5. Edit your converted Amazon Redshift RSQL scripts or save them. For more information, see **Editing and saving your converted FastExport job scripts** (p. 205).

### Managing Teradata FastExport job scripts with AWS SCT

You can add multiple Teradata FastExport job scripts or remove a FastExport job script from your AWS SCT project.

**To add a new FastExport job script to your AWS SCT project**

1. Expand the **Scripts** node in the left panel.
2. Choose the **FastExport** node, and open the context (right-click) menu.
3. Choose **Load scripts**.
4. Enter the information that is required to add a new FastExport job script and configure substitution variables. For more information, see **Adding FastExport job scripts to your AWS SCT project** (p. 202) and **Configuring substitution variables in FastExport job scripts** (p. 202).

**To remove a FastExport job script from your AWS SCT project**

1. Expand the **FastExport** node under **Scripts** in the left panel.
2. Choose the script to remove, and open the context (right-click) menu.
3. Choose **Delete script**.

### Creating an assessment report for a Teradata FastExport job script conversion with AWS SCT

The **FastExport job script conversion assessment report** provides information about converting the FastExport commands and SQL statements from your FastExport scripts to a format compatible with Amazon Redshift RSQL. The assessment report includes action items for FastExport commands and SQL statements that AWS SCT can’t convert.

**To create a script conversion assessment report for a Teradata FastExport job**

1. Expand the **FastExport** node under **Scripts** in the left panel.
2. Choose the script to convert, open the context (right-click) menu, and then choose **Create report**.
3. View the **Summary** tab. The **Summary** tab displays the executive summary information from the FastExport job script assessment report. It includes conversion results for all FastExport commands and SQL statements from your source scripts.
4. You can save a local copy of the FastExport job script conversion assessment report as either a PDF file or a comma-separated values (CSV) file.
a. To save the FastExport job script conversion assessment report as a PDF file, choose **Save to PDF** at upper right.

The PDF file contains the executive summary, action items, and recommendations for scripts conversion.

b. To save the FastExport job script conversion assessment report as a CSV file, choose **Save to CSV** at upper right.

The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the scripts.

5. Choose the **Action items** tab. This tab contains a list of items that require manual conversion to Amazon Redshift RSQL. When you select an action item from the list, AWS SCT highlights the item from your source FastExport job script that the action item applies to.

### Editing and saving your converted Teradata FastExport job scripts with AWS SCT

You can edit your converted scripts in the lower panel of your AWS SCT project. AWS SCT stores the edited script as part of your project.

**To save your converted scripts**

1. Expand the **RSQL scripts** node under **Scripts** in the target database panel.
2. Choose your converted script, open the context (right-click) menu, and choose **Save script**.
3. Enter the path to the folder to save the converted script and choose **Save**.

AWS SCT saves the converted script to a file and opens this file.

### Converting Teradata FastLoad job scripts to Amazon Redshift RSQL with AWS SCT

You can use the AWS Schema Conversion Tool (AWS SCT) to convert Teradata FastLoad job scripts to Amazon Redshift RSQL.

A **Teradata FastLoad script** is a set of commands that use multiple sessions to load data in an empty table on a Teradata Database. Teradata FastLoad processes a series of Teradata FastLoad commands and SQL statements. The Teradata FastLoad commands provide session control and data handling of the data transfers. The SQL statements create, maintain, and drop tables.

AWS SCT converts Teradata FastLoad commands and SQL statements to a format compatible with Amazon Redshift RSQL. After you migrate the Teradata database to Amazon Redshift, you can use these converted scripts to load data to your Amazon Redshift database.

### Adding FastLoad job scripts to your AWS SCT project

You can add multiple scripts to a single AWS SCT project.

**To add a FastLoad job script to your AWS SCT project**

1. Create a new project in AWS SCT, or open an existing project. For more information, see the section called “Creating a project” (p. 16).
2. Choose **Add source** from the menu, and then choose **Teradata** to add your source database to the project. For more information, see Using Teradata as a source (p. 117).

3. Choose **Add target** from the menu and add a target Amazon Redshift database to your AWS SCT project.

   You can use a virtual Amazon Redshift target database platform. For more information, see Using virtual targets (p. 123).

4. Create a new mapping rule that includes your source Teradata database and your Amazon Redshift target. For more information, see Adding a new mapping rule (p. 122).

5. On the **View** menu, choose **Main view**.

6. In the left panel, expand the **Scripts** node.

7. Choose **FastLoad**, open the context (right-click) menu, and then choose **Load scripts**.

8. Enter the location of your source Teradata FastLoad job scripts and choose **Select folder**.

   AWS SCT displays the **Load scripts** window.

9. Do one of the following:

   - If your Teradata FastLoad job scripts don't include the substitution variables, choose **No substitution variables**, and then choose **OK** to add scripts to your AWS SCT project.
   - If your Teradata FastLoad job scripts include the substitution variables, configure the substitution variables. For more information, see Configuring substitution variables in FastLoad job scripts (p. 206).

### Configuring substitution variables in Teradata FastLoad job scripts with AWS SCT

Your Teradata FastLoad job scripts might include substitution variables. For example, you can use a single script with substitution variables to load data to different databases.

Before you run a FastLoad job script with substitution variables, make sure to assign the values for all variables. To do this, you can use other tools or applications such as a Bash script, UC4 (Automic), and so on.

AWS SCT can resolve and convert substitution variables only after you assign their values. Before you start the conversion of your source Teradata FastLoad job scripts, make sure that you assign values for all substitution variables. You can use AWS SCT to configure substitution variables in your Teradata scripts.

#### To configure substitution variables in your FastLoad job script

1. When you add your source Teradata FastLoad job scripts to your AWS SCT project, choose **Substitution variables are used**. For more information about adding these scripts, see Adding FastLoad job scripts to your AWS SCT project (p. 205).

2. For **Define variable format**, enter a regular expression that matches all substitution variables in your script.

   For example, if the names of your substitution variables start with `_${` and end with `}_`, use the `\${\w+}` regular expression. To match substitution variables that start either with a dollar sign or a percent sign, use the `\$\w+|\%\w+` regular expression.

   Regular expressions in AWS SCT conform to the Java regular expression syntax. For more information, see java.util.regex Class Pattern in the Java documentation.

3. Choose **OK** to load scripts to your AWS SCT project, and then choose **OK** to close the **Load scripts** window.
4. In the left panel, expand the **Scripts** node. Choose **FastLoad**, and then choose your folder with scripts. Open the context (right-click) menu, and then choose **Export variables** under **Substitution variables**.

Also, you can export substitution variables for one script. Expand your folder with scripts, choose your script, open the context (right-click) menu, and choose **Export variables** under **Substitution variables**.

5. Enter the name of the comma-separated value (CSV) file to save the substitution variables, and then choose **Save**.

6. Open this CSV file and fill in the values for the substitution variables.

   Depending on the operating system, AWS SCT uses different formats for the CSV file. The values in the file might be either enclosed in quotation marks or not. Make sure that you use the same format for the values of substitution variables as the other values in the file. AWS SCT can’t import the CSV file with values in different formats.

7. Save the CSV file.

8. In the left panel, expand the **Scripts** node. Choose **FastLoad**, and then choose your script. Open the context (right-click) menu, and then choose **Import variables** under **Substitution variables**.

9. Choose your CSV file, and then choose **Open**.

10. Choose **Variables** to view all discovered substitution variables and their values.

### Converting Teradata FastLoad job scripts with AWS SCT

Following, find how to convert Teradata FastLoad job to Amazon Redshift RSQL using AWS SCT.

**To convert a Teradata FastLoad job script to Amazon Redshift RSQL**

1. Add your FastLoad job scripts to your AWS SCT project. For more information, see *Adding FastLoad job scripts to your AWS SCT project* (p. 205).

2. Configure the substitution variables. For more information, see *Configuring substitution variables in FastLoad job scripts* (p. 206).

3. In the left panel, expand the **Scripts** node.

4. Do one of the following:

   - To convert a single FastLoad job script, expand the **FastLoad** node, choose the script to convert, and then choose **Convert script** from the context (right-click) menu.
   - To covert multiple scripts, make sure that you select all scripts to convert. Choose **FastLoad**, open the context (right-click) menu, and then choose **Convert script**. Then do one of the following:

     - If you store your source data file on Amazon S3, choose **S3 object path** for **Source data file location**.

       Enter values for **Amazon S3 bucket folder** and **Amazon S3 bucket for manifest file** for your source data file.

     - If you don't store your source data file on Amazon S3, choose **Host address** for **Source data file location**.

       Enter values for **URL or IP address of the host**, **Host user login name**, and **Amazon S3 bucket for manifest file** for your source data file.

5. Choose **OK**.

AWS SCT converts all your selected Teradata FastLoad job scripts to a format compatible with Amazon Redshift RSQL. Find your converted scripts in the **Scripts** node in the target database panel.
6. Edit your converted Amazon Redshift RSQL scripts or save them. For more information, see Editing and saving your converted FastLoad job scripts (p. 209).

**Managing Teradata FastLoad job scripts with AWS SCT**

You can add multiple Teradata FastLoad job scripts or remove a FastLoad job script from your AWS SCT project.

**To add a new FastLoad job script to your AWS SCT project**

1. Expand the Scripts node in the left panel.
2. Choose the FastLoad node and open the context (right-click) menu.
3. Choose Load scripts.
4. Enter the information that is required to add a new FastLoad job script and configure substitution variables. For more information, see Adding FastLoad job scripts to your AWS SCT project (p. 205) and Configuring substitution variables in FastLoad job scripts (p. 206).

**To remove a FastLoad job script from your AWS SCT project**

1. Expand the FastLoad node under Scripts in the left panel.
2. Choose the script to remove, and open the context (right-click) menu.
3. Choose Delete script.

**Creating an assessment report for a Teradata FastLoad job script conversion with AWS SCT**

The FastLoad job script conversion assessment report provides information about converting the FastLoad commands and SQL statements. The conversion is from your source scripts to a format compatible with Amazon Redshift RSQL. The assessment report includes action items for FastLoad commands and SQL statements that AWS SCT can’t convert.

**To create a script conversion assessment report for a Teradata FastLoad job**

1. Expand the FastLoad node under Scripts in the left panel.
2. Choose the script to convert, open the context (right-click) menu, and then choose Create report.
3. View the Summary tab.

   The Summary tab displays the executive summary information from the FastLoad job script assessment report. It includes conversion results for all FastLoad commands and SQL statements from your source scripts.
4. (Optional) Save a local copy of the FastLoad job script conversion assessment report as either a PDF file or a comma-separated value (CSV) file:
   - To save the FastLoad job script conversion assessment report as a PDF file, choose Save to PDF at upper right.
     The PDF file contains the executive summary, action items, and recommendations for script conversion.
   - To save the FastLoad job script conversion assessment report as a CSV file, choose Save to CSV at upper right.
The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the scripts.

5. Choose the **Action items** tab. This tab contains a list of items that require manual conversion to Amazon Redshift RSQL. When you select an action item from the list, AWS SCT highlights the item from your source FastLoad job script that the action item applies to.

**Editing and saving your converted Teradata FastLoad job scripts with AWS SCT**

You can edit your converted scripts in the lower panel of your AWS SCT project. AWS SCT stores the edited script as part of your project.

**To save your converted scripts**

1. Expand the **RSQL scripts** node under **Scripts** in the target database panel.
2. Choose your converted script, open the context (right-click) menu, and choose **Save script**.
3. Enter the path to the folder to save the converted script and choose **Save**.

AWS SCT saves the converted script to a file and opens this file.

**Converting Teradata MultiLoad job scripts to Amazon Redshift RSQL with AWS SCT**

You can use AWS SCT to convert Teradata MultiLoad job scripts to Amazon Redshift RSQL.

A **Teradata MultiLoad job script** is a set of commands for batch maintenance of your Teradata Database. A Teradata MultiLoad import task performs a number of different insert, update, and delete operations on up to five different tables and views. Teradata MultiLoad delete tasks can remove large numbers of rows from a single table.

AWS SCT converts Teradata MultiLoad commands and SQL statements to a format compatible with Amazon Redshift RSQL. After you migrate the Teradata database to Amazon Redshift, use these converted scripts to manage data in your Amazon Redshift database.

**Adding MultiLoad job scripts to your AWS SCT project**

You can add multiple scripts to a single AWS SCT project.

**To add a MultiLoad job script to your AWS SCT project**

1. Create a new project in AWS SCT or open an existing project. For more information, see the section called “Creating a project” (p. 16).
2. Choose **Add source** from the menu, and then choose **Teradata** to add your source database to the project. For more information, see Using Teradata as a source (p. 117).
3. Choose **Add target** from the menu and to add a target Amazon Redshift database to your AWS SCT project.

You can use a virtual Amazon Redshift target database platform. For more information, see Using virtual targets (p. 123).
4. Create a new mapping rule that includes your source Teradata database and your Amazon Redshift target. For more information, see Adding a new mapping rule (p. 122).
5. On the View menu, choose Main view.
6. In the left panel, expand the Scripts node.
7. Choose MultiLoad, open the context (right-click) menu, and then choose Load scripts.
8. Enter the location of your source Teradata MultiLoad job scripts and choose Select folder.
   AWS SCT displays the Load scripts window.
9. Do one of the following:
   - If your Teradata MultiLoad job scripts don't include the substitution variables, choose No substitution variables, and then choose OK to add scripts to your AWS SCT project.
   - If your Teradata MultiLoad job scripts include the substitution variables, configure the substitution variables. For more information, see Configuring substitution variables in MultiLoad job scripts (p. 210).

**Configuring substitution variables in Teradata MultiLoad job scripts with AWS SCT**

Your Teradata MultiLoad job scripts might include substitution variables. For example, you can use a single script with substitution variables to load data to different databases.

Before you run a MultiLoad job script with substitution variables, make sure to assign the values for all variables. To do this, you can use other tools or applications such as a Bash script, UC4 (Automic), and so on.

AWS SCT can resolve and convert substitution variables only after you assign their values. Before you start the conversion of your source Teradata MultiLoad job scripts, make sure that you assigned values for all substitution variables. You can use AWS SCT to configure substitution variables in your Teradata scripts.

**To configure substitution variables in your MultiLoad job script**

1. When you add your source Teradata MultiLoad job scripts to your AWS SCT project, choose Substitution variables are used. For more information about adding these scripts, see Adding MultiLoad job scripts to your AWS SCT project (p. 209).
2. For Define variable format, enter a regular expression that matches all substitution variables in your script.
   - For example, if the names of your substitution variables start with $ ( and end with ), use the \$ \{w+\} regular expression. To match substitution variables that start either with a dollar sign or a percent sign, use the \$\\w+|\\%\\w+ regular expression.

   Regular expressions in AWS SCT conform to the Java regular expression syntax. For more information, see java.util.regex Class Pattern in the Java documentation.
3. Choose OK to load scripts to your AWS SCT project, and then choose OK to close the Load scripts window.
4. Choose Variables to view all discovered substitution variables and their values.
5. For Value, enter the value for the substitution variable.

**Converting Teradata MultiLoad job scripts with AWS SCT**

Following, find how to convert Teradata MultiLoad job to Amazon Redshift RSQL using AWS SCT.
Managing MultiLoad job scripts

To convert a Teradata MultiLoad job script to Amazon Redshift RSQL

1. Add your MultiLoad job scripts to your AWS SCT project. For more information, see Adding MultiLoad job scripts to your AWS SCT project (p. 209).
2. Configure the substitution variables and enter their values. For more information, see Configuring substitution variables in MultiLoad job scripts (p. 210).
3. In the left panel, expand the Scripts node.
4. Do one of the following:
   - To convert a single MultiLoad job script, expand the MultiLoad node, choose the script to convert, and then choose Convert script from the context (right-click) menu.
   - To covert multiple scripts, make sure that you select all scripts to convert. Choose MultiLoad, open the context (right-click) menu, and then choose Convert script.
5. Do one of the following:
   - If you store your source data file on Amazon S3, choose S3 object path for Source data file location.
     Enter Amazon S3 bucket folder and Amazon S3 bucket for manifest file for your source data file.
   - If you don't store your source data file on Amazon S3, choose Host address for Source data file location.
     Enter URL or IP address of the host, Host user login name, and Amazon S3 bucket for manifest file for your source data file.
6. Choose OK.

AWS SCT converts all your selected Teradata MultiLoad job scripts to a format compatible with Amazon Redshift RSQL. Find your converted scripts in the Scripts node in the target database panel.
7. Edit your converted Amazon Redshift RSQL scripts or save them. For more information, see Editing and saving your converted MultiLoad job scripts (p. 212).

Managing Teradata MultiLoad job scripts with AWS SCT

You can add multiple Teradata MultiLoad job scripts or remove a MultiLoad job script from your AWS SCT project.

To add a new MultiLoad job script to your AWS SCT project

1. Expand the Scripts node in the left panel.
2. Choose the MultiLoad node and open the context (right-click) menu.
3. Choose Load scripts.
4. Enter the information that is required to add a new MultiLoad job script and configure substitution variables. For more information, see Adding MultiLoad job scripts to your AWS SCT project (p. 209) and Configuring substitution variables in MultiLoad job scripts (p. 210).

To remove a MultiLoad job script from your AWS SCT project

1. Expand the MultiLoad node under Scripts in the left panel.
2. Choose the script to remove, and open the context (right-click) menu.
3. Choose Delete script.
Creating an assessment report for a Teradata MultiLoad job script conversion with AWS SCT

The MultiLoad job script conversion assessment report provides information about converting the MultiLoad commands and SQL statements. The conversion is from your source scripts to Amazon Redshift RSQL commands and SQL statements for Amazon Redshift. The assessment report includes action items for MultiLoad commands and SQL statements that AWS SCT can't convert.

To create a script conversion assessment report for a Teradata MultiLoad job

1. Expand the MultiLoad node under Scripts in the left panel.
2. Choose the scripts to create the assessment report for, open the context (right-click) menu, and then choose Create report.
3. View the Summary tab. The Summary tab displays the executive summary information from the MultiLoad job script assessment report. It includes conversion results for all MultiLoad commands and SQL statements from your source scripts.
4. (Optional) Save a local copy of the MultiLoad job script conversion assessment report as either a PDF file or comma-separated value (CSV) files:
   - To save the MultiLoad job script conversion assessment report as a PDF file, choose Save to PDF at upper right. The PDF file contains the executive summary, action items, and recommendations for scripts conversion.
   - To save the MultiLoad job script conversion assessment report as CSV files, choose Save to CSV at upper right. AWS SCT creates two CSV files. These files contain the executive summary, action items, recommended actions, and an estimated complexity of manual effort required to convert the scripts.
5. Choose the Action items tab. This tab contains a list of items that require manual conversion to Amazon Redshift RSQL. When you select an action item from the list, AWS SCT highlights the item from your source MultiLoad job script that the action item applies to.

Editing and saving your converted Teradata MultiLoad job scripts with AWS SCT

You can edit your converted scripts in the lower panel of your AWS SCT project. AWS SCT stores the edited script as part of your project.

To save your converted scripts

1. Expand the RSQL scripts node under Scripts in the target database panel.
2. Choose your converted script, open the context (right-click) menu, and choose Save script.
3. Enter the path to the folder to save the converted script and choose Save.

AWS SCT saves the converted script to a file and opens this file.
Using AWS SCT with AWS DMS

Using an AWS SCT replication agent with AWS DMS

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

When working with AWS Snowball Edge, the AWS SCT agent replicates data to the AWS 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 Using data extraction agents (p. 215).

Using an AWS SCT data extraction agent with AWS DMS

In AWS SCT, you can find a data extraction agent (aws-schema-conversion-tool-extractor) that helps make migrations from Apache Cassandra to Amazon DynamoDB easier. Cassandra and DynamoDB are NoSQL databases, but they differ in system architecture and data representation. You can use wizard-based workflows in AWS SCT to automate the Cassandra-to-DynamoDB migration process. AWS SCT integrates with AWS Database Migration Service (AWS DMS) to perform the actual migration.

For more information, see Using data extraction agents (p. 215).

Increasing logging levels when using AWS SCT with AWS DMS

You can increase logging levels when using AWS SCT with AWS DMS, for example if you need to work with AWS Support.

After installing AWS SCT and the required drivers, open the application by choosing the AWS SCT icon. If you see an update notification, you can choose to update before or after your project is complete. If an auto-project window opens, close the window and manually create a project.

To increase logging levels when using AWS SCT with AWS DMS

2. In the Global settings window, choose Logging.
3. For Debug mode, choose True.
4. From the Message level section, you can modify the following types of logs:
• General
• Loader
• Parser
• Printer
• Resolver
• Telemetry
• Converter

By default, all message levels are set to **Info**.

5. Choose a level of logging for any message level types that you want to change:
   • Trace (most detailed logging)
   • Debug
   • Info
   • Warning
   • Error (least detailed logging)
   • Critical
   • Mandatory

6. Choose **Apply** to modify settings for your project.
7. Choose **OK** to close the **Global settings** window.
Using data extraction agents

In some migration scenarios, the source and target databases are very different from one another and require additional data transformation. AWS Schema Conversion Tool (AWS SCT) is extensible, so that you can address these scenarios using an agent. An agent is an external program that's integrated with AWS SCT, but performs data transformation elsewhere (such as on an Amazon EC2 instance). In addition, an AWS SCT agent can interact with other AWS services on your behalf, such as creating and managing AWS Database Migration Service tasks for you.

Topics
- Extracting data from on-premises databases using data extraction agents (p. 215)
- Migrating data from on-premises databases using AWS DMS tasks (p. 219)
- Migrating data from an on-premises data warehouse to Amazon Redshift (p. 220)
- Migrating data from Apache Cassandra to Amazon DynamoDB (p. 249)

Extracting data from on-premises databases using data extraction agents

You can use an AWS SCT agent to extract data from your on-premises relational database. You can then migrate the extracted data to the AWS Cloud.

Installing extraction agents

You can install multiple data extraction agents on individual computers, separate from the computer that is running AWS SCT. Use this approach to run multiple data extraction agents in parallel.

The AWS DMS data extraction agent is provided as part of AWS SCT installation package. Find the installation file in the dmsagent folder of the AWS SCT installation directory. For more information, see Installing, verifying, and updating (p. 5).

To install an AWS DMS data extraction agent

1. Make sure that AWS SCT and the AWS DMS agent are installed on separate machines. Make sure that the AWS DMS agent is installed on the same machine as the Open Database Connectivity (ODBC) drivers and, as needed, the AWS Snowball Edge client.
2. In the dmsagent folder of the AWS SCT installation directory, locate the RPM Package Manager (RPM) file called aws-schema-conversion-tool-dms-agent-X.X-XX.x86_64.rpm.

   Copy it to the AWS DMS agent machine.
3. On the agent machine, run the following command to install the DMS agent. To simplify permissions, run this command as the root user.

   ```bash
   sudo rpm -i aws-schema-conversion-tool-dms-agent-X.X-XX.x86_64.rpm
   ```

   This command uses the default installation location of /opt/amazon/aws-schema-conversion-tool-dms-agent. To install the DMS agent to a different location, use the following option.
Installing agents

4. Run the following command to verify that the data extraction agent is running.

   `ps -ef | grep repctl`

   The output of this command should show two processes running.

5. Configure the data extraction agent as follows:
   a. Run the `configure.sh` script.

   ```
   sudo /opt/amazon/aws-schema-conversion-tool-dms-agent/bin/configure.sh
   ```

   A password prompt appears.

   b. Enter a password. Make sure that your password is 8–20 alphanumeric characters, with at least one digit and one uppercase character. When prompted, enter the password again to confirm it. You use the password later to register the DMS agent with AWS SCT, so keep it handy.

   A port prompt appears.

   c. Provide a port number. Choose an unused port number for the data extraction agent to listen on for AWS SCT connections. The default is 3554. You might have to configure your firewall to allow connectivity.

   The output is as follows, confirming that the service is started.

   ```
   Starting service...
   /opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
   /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version information available (required by /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/gssapi_krb5.so.2)
   /opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
   /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version information available (required by /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/krb5.so.3)
   AWS Schema Conversion Tool DMS Agent was sent a stop signal
   AWS Schema Conversion Tool DMS Agent is no longer running
   [service command] Succeeded
   /opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
   /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version information available (required by /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/gssapi_krb5.so.2)
   /opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
   /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version information available (required by /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/krb5.so.3)
   AWS Schema Conversion Tool DMS Agent was started as PID 1608
   ```

   We recommend that you install the AWS Command Line Interface (AWS CLI). Using the AWS CLI, you can check the AWS Snowball Edge to see the data files written to the device. You use the AWS credentials retrieved from the AWS Snowball Edge to access the AWS Snowball Edge device. For example, you might run the following command.

   ```
   aws s3 ls --profile SnowballEdge --endpoint https://192.0.2.0 :8080 bucket-name --recursive
   ```

   This command produces the following output.
To start the data extraction agent

- Run the following command in the /opt/amazon/aws-schema-conversion-tool-dms-agent/bin directory.

  ./aws-schema-conversion-tool-dms-agent start

To stop the data extraction agent

- Run the following command in the /opt/amazon/aws-schema-conversion-tool-dms-agent/bin directory.

  ./aws-schema-conversion-tool-dms-agent stop

Registering data extraction agents

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 database.

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

To register a data extraction agent

1. Create a new project in AWS SCT or open an existing project. For more information, see Creating an AWS SCT project (p. 16).
2. Add your source and target databases, and create a new mapping rule. For more information, see Adding a new mapping rule (p. 122).
3. On the View menu, choose Data Migration view (other).
4. Choose Register, and then choose DMS data agent.
   
   If your mapping rules imply using only one type of agent, the New Agent Registration dialog box appears automatically after you choose Register.
5. Enter your information in the New Agent Registration dialog box.

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enter the name of the agent.</td>
</tr>
<tr>
<td>Host name</td>
<td>Enter the IP address of the machine where you installed the data extraction agent.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the port number that you used when you configured the data extraction agent.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password that you used when you configured the data extraction agent.</td>
</tr>
</tbody>
</table>

6. Choose Register to register the agent with your AWS SCT project.
Creating, running, and monitoring an AWS SCT data extraction task

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

**To assign tasks to agents and migrate data**

1. In AWS SCT, 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 database.
2. Open the context (right-click) menu for each table, and then choose **Create task**. The Create Local task dialog box opens.
3. For **Task name**, enter a name for the task.
4. For **Migration mode**, choose **Extract only** to extract your data, and save the data to your local working folders.
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. (Optional) Choose **Enable task logging** to see detailed information about a task. You can use the task log to debug problems.
   
   If you turn on 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 that 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 a task**

1. In AWS SCT, for **View**, choose **Data Migration view (standard DMS)**. The **Agents** tab appears.
2. Choose the **Tasks** tab. Your tasks appear in the grid at the top. 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 that you chose, you see the task divided into **Extract**, **Upload**, and **Copy**.
4. Choose **Start** for a task to start that task.

If you turned on logging when you set up your task, you can monitor the status of your task while it works. The subtasks run in parallel. The extract, upload, and copy also run in parallel.

**To monitor a task**

- View the log as follows:
  
  a. Choose **Download log**.
A message appears with the name of the folder that contains the log file. Dismiss the message. A link appears in the Task details tab.

b. Choose the link to open the folder that contains the log file.

If you close AWS SCT, 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.

You can save data extraction tasks to your local disk and restore them to the same or another project by using export and import.

To export a task, make sure that you have at least one extraction task created in a project. Then choose the vertical ellipsis (⋮) and choose Export task or Export all tasks.

When you export one extraction task, AWS SCT creates a separate .xml file for that task. The .xml file stores that task’s metadata information, such as task properties, description, and subtasks. The .xml file doesn’t contain information about processing of an extraction task.

When you export all extraction tasks, AWS SCT creates a separate .xml file for each task.

You can import a single extraction task or all of the tasks created in the project. The following information is recreated when the task is imported:

- Task progress
- Subtask and stage states
- Distribution of extracting agents by subtasks and stages
- Task and subtask IDs
- Task name

To import a task, choose the vertical ellipsis (⋮) and choose Import task.

Migrating data from on-premises databases using AWS DMS tasks

You can use an AWS Database Migration Service (AWS DMS) task to migrate data from your on-premises relational database to the AWS Cloud. You can create, run, and monitor these AWS DMS tasks from AWS SCT.

Creating AWS DMS tasks from AWS SCT

Use the following procedure to create an AWS DMS task in AWS SCT.

To create a data migration task

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

   For performance reasons, AWS recommends that you create multiple tasks for multiple tables.

2. Open the context (right-click) menu for your database schema, which includes tables to migrate, and then choose Create DMS task.

   The Create DMS task dialog box opens.
3. For **Task name**, enter a name for the task.
4. For **Replication instance**, choose your replication instance from the AWS DMS console.
   Make sure that you configured the replication instance in the AWS DMS console for the same AWS profile that you use in AWS SCT. Also, make sure that the major version of the AWS DMS task matches the version of the replication instance.
5. For **Source endpoint**, choose the source endpoint for data migration.
6. For **Target endpoint**, choose the target endpoint for data migration.
7. For **Migration type**, choose one of the following:
   - **Migrate existing data** – Migrate your data to the target database.
   - **Migrate existing data and replicate ongoing changes** – Migrate your data to the target database and set up ongoing replication or change data capture (CDC).
   - **Replicate data changes only** – Set up ongoing replication or CDC.
8. For **Target table preparation mode**, choose one of the following:
   - **Do nothing** – Leave data in the target database tables.
   - **Drop tables on target** – Drop tables in the target database and create new tables before data migration.
   - **Truncate** – Remove data from the target database tables before data migration.
9. For **Include LOB columns in replication**, choose one of the following:
   - **Don't include LOB columns** – Skip large object (LOB) columns during data migration.
   - **Full LOB mode** – Migrate LOB columns. For this option, enter the LOB chunk size in KB.
   - **Limited LOB mode** – Migrate LOB columns. For this option, enter the maximum size of LOB in KB.
10. For **Logging**, choose **Enable** to see detailed information about your task. You can use the task log to debug problems.
11. Choose **Create** to create the task.

Or you can create and configure your data migration task in the AWS DMS console. For more information, see Creating a task in the AWS Database Migration Service User Guide.

**Managing AWS DMS tasks from AWS SCT**

Use the following procedures to manage AWS DMS tasks in AWS SCT.

**To manage data migration tasks**

1. On the **View** menu, choose **Data migration view (standard DMS)**.
2. On the **Tasks** tab, review your data migration tasks. AWS SCT displays the status of your AWS DMS tasks in the top grid, and the status of subtasks in the bottom grid.
3. Choose a task in the top grid and expand it.
4. Choose **Start** for a task to start that task.

**Migrating data from an on-premises data warehouse to Amazon Redshift**

You can use an AWS SCT agent to extract data from your on-premises data warehouse and migrate it to Amazon Redshift. The agent extracts your data and uploads the data to either Amazon S3 or, for large-
Large-scale migrations

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 Edge 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 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 AWS Snowball Edge devices.

When you use AWS SCT and an AWS Snowball Edge device, you migrate your data in two stages. First, you use AWS SCT to process the data locally and then move that data to the AWS Snowball Edge device. You then send the device to AWS using the AWS Snowball Edge 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)
Prerequisite settings for using 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 Storing AWS service profiles in the AWS SCT (p. 34).

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 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 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:

- 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 AWS SCT (p. 5) to download it. The .zip file that contains the AWS SCT installer file also contains the extraction agent installer file.

2. Download and install the latest version of Amazon Corretto 11. For more information, see Downloads for Amazon Corretto 11 in the **Amazon Corretto 11 User Guide**.

3. 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>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>

4. Install the extraction agent on a separate computer by copying the installer file to the new computer.

5. 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>Microsoft Windows</td>
<td>Double-click the file to run the installer.</td>
</tr>
<tr>
<td>RHEL</td>
<td>Run the following commands in the folder that you downloaded or moved the file to.</td>
</tr>
</tbody>
</table>
Operating system | Installation instructions
--- | ---
 | sudo rpm -ivh aws-schema-conversion-tool-extractor-1.0.build-number.x86_64.rpm
 | sudo ./sct-extractor-setup.sh --config
Ubuntu Linux | Run the following commands in the folder that you downloaded or moved the file to.
 | sudo dpkg -i aws-schema-conversion-tool-extractor-1.0.build-number.deb
 | sudo ./sct-extractor-setup.sh --config

6. Choose **Next**, accept the license agreement, and choose **Next**.
7. Enter the path to install the AWS SCT data extraction agent, and choose **Next**.
8. Choose **Install** to install your data extraction agent.

AWS SCT installs your data extraction agent. To complete the installation, configure your data extraction agent. AWS SCT automatically launches the configuration setup program. For more information, see Configuring extraction agents (p. 225).

9. Choose **Finish** to close the installation wizard after you configure your data extraction agent.

## 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**

1. Launch the configuration setup program:
   - On Microsoft Windows, AWS SCT launches the configuration setup program automatically during the installation of a data extraction agent.
   
   As needed, launch the setup program manually on Windows using the following command.

   ```
   cd path/AWSSchemaConversionTool-Extractor.jar
   java -jar AWSSchemaConversionTool-Extractor.jar -config
   ```

   In the preceding example, `path` is the path where you installed the AWS SCT data extraction agent.
   - On RHEL and Ubuntu, run the `sct-extractor-setup.sh` file from the location where you installed the agent.

   The setup program prompts you for information. For each prompt, a default value appears.

2. Accept the default value at each prompt, or enter a new value.

Specify the following information:

- For **Listening port**, enter the port number the agent listens on.
- For **Add a source vendor**, enter yes, and then enter your source data warehouse platform.
- For **JDBC driver**, enter the location where you installed the JDBC drivers.
Installing agents

- For **Working folder**, enter the path where the AWS SCT data extraction agent will store the extracted data. 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.
- For **Enable SSL communication**, enter *yes*.
- For **Key store**, enter the location of the key store file.
- For **Key store password**, enter the password for the key store.
- For **Enable client SSL authentication**, enter *yes*.
- For **Trust store**, enter the location of the trust store file.
- For **Trust store password**, enter 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.

```bash
$ cat settings.properties
#extractor.start.fetch.size=20000
#extractor.out.file.size=10485760
#extractor.source.connection.pool.size=20
#extractor.source.connection.pool.min.evictable.idle.time.millis=30000
#extractor.extracting.thread.pool.size=10
vendor=TERADATA
driver.jars=/usr/share/lib/jdbc/terajdbc4.jar
port=8192
redshift.driver.jars=/usr/share/lib/jdbc/RedshiftJDBC42-1.2.43.1067.jar
working.folder=/data/sct
extractor.private.folder=/home/ubuntu
ssl.option=OFF
```

To change configuration settings, you can edit the `settings.properties` file using a text editor or run the agent configuration again.

**Installing and configuring extraction agents with dedicated copying agents**

You can install extraction agents in a configuration that has shared storage and a dedicated copying agent. The following diagram illustrates this scenario.
That configuration can be useful when a source database server supports up to 120 connections, and your network has ample storage attached. Use the procedure following to configure extraction agents that have a dedicated copying agent.

**To install and configure extraction agents and a dedicated copying agent**

1. Make sure that the working directory of all extracting agents uses the same folder on shared storage.
2. Install extractor agents by following the steps in Installing extraction agents (p. 224).
3. Configure extraction agents by following the steps in Configuring extraction agents (p. 225), but specify only the source JDBC driver.
4. Configure a dedicated copying agent by following the steps in Configuring extraction agents (p. 225), but specify only an Amazon Redshift JDBC driver.

**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>Microsoft Windows</td>
<td>Double-click the <code>StartAgent.bat</code> 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 (other). The Agents tab appears. If you have previously registered agents, AWS SCT displays them in a grid at the top of the tab.
3. Choose Register.

   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. Choose Redshift data agent, and then choose OK.
5. Enter your information on the Connection tab of the dialog box:
   a. For Description, enter a description of the agent.
   b. For Host Name, enter the host name or IP address of the computer of the agent.
   c. For Port, enter the port number that the agent is listening on.
   d. Choose Register to register the agent with your AWS SCT project.
6. 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 starts, it can use this path to find the seed.dat file to access the key-trust store information for the database it acts on.

You might need to recover passwords that an agent has stored in these cases:

- If the user loses the seed.dat file and the AWS SCT agent's location and port didn't change.
- If the user loses the seed.dat file and the AWS SCT agent's location and port has changed. In this case, the change usually occurs because the agent was migrated to another host or port and the information in the seed.dat file is no longer valid.

In these cases, if an agent is started without SSL, it starts and then accesses the previously created agent storage. It then goes to the Waiting for recovery state.

However, in these cases, if an agent is started with SSL you can't restart it. This is because the agent can't decrypt the passwords to certificates stored in the settings.properties file. In this type of startup, the agent fails to start. An error similar to the following is written in the log: “The agent could not start with SSL mode enabled. Please reconfigure the agent. Reason: The password for keystore is incorrect.”

To fix this, create a new agent and configure the agent to use the existing passwords for accessing the SSL certificates. To do so, use the following procedure.

After you perform this procedure, the agent should run and go to the Waiting for recovery state. AWS SCT automatically sends the needed passwords to an agent in the Waiting for recovery state. When the agent has the passwords, it restarts any tasks. No further user action is required on the AWS SCT side.
To reconfigure the agent and restore passwords for accessing SSL certificates

1. Install a new AWS SCT agent and run configuration.
2. Change the `agent.name` property in the `instance.properties` file to the name of the agent the storage was created for, to have the new agent work with existing agent storage.

   The `instance.properties` file is stored in the agent's private folder, which is named using the following convention: `{output.folder}\dmt\{hostName}\{portNumber}\.
3. Change the name of `{output.folder}` to that of the previous agent's output folder.

   At this point, AWS SCT is still trying to access the old extractor at the old host and port. As a result, the inaccessible extractor gets the status FAILED. You can then change the host and port.
4. Modify the host, port, or both for the old agent by using the Modify command to redirect the request flow to the new agent.

When AWS SCT can ping the new agent, AWS SCT receives the status Waiting for recovery from the agent. AWS SCT then automatically recovers the passwords for the agent.

Each agent that works with the agent storage updates a special file called `storage.lck` located at `{output.folder}\{agentName}\storage`. This file contains the agent's network ID and the time until which the storage is locked. When the agent works with the agent storage, it updates the `storage.lck` file and extends the lease of the storage by 10 minutes every 5 minutes. No other instance can work with this agent storage before the lease expires.

Creating data migration rules in AWS SCT

Before you extract your data with the AWS Schema Conversion Tool, you can set up filters that reduce the amount of data that you extract. You can create data migration rules by using WHERE 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 migration rules and save the filters as part of your project. With your project open, use the following procedure to create data migration rules.

To create data migration rules

1. Open the View menu, and then choose Data Migration view (other).
2. Choose Data migration rules, and then choose Add new rule.
3. Configure your data migration rule:
   a. For Name, enter a name for your data migration rule.
   b. For Where schema name is like, enter a filter to apply to schemas. In this filter, a WHERE clause is evaluated by using a LIKE clause. To choose one schema, enter an exact schema name. To choose multiple schemas, use the “%” character as a wildcard to match any number of characters in the schema name.
   c. For table name like, enter a filter to apply to tables. In this filter, a WHERE clause is evaluated by using a LIKE clause. To choose one table, enter an exact name. To choose multiple tables, use the “%” character as a wildcard to match any number of characters in the table name.
   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.
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.

### Changing extractor and copy settings from project settings

From the **Project settings** window in AWS SCT, you can choose settings for data extraction agents and the Amazon Redshift COPY command.

To choose these settings, choose **Settings**, **Project settings**, and then choose **Data migration**. Here, you can edit **Extraction settings**, **Amazon S3 settings**, and **Copy settings**.

Use the instructions in the following table to provide the information for **Extraction settings**.

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression format</td>
<td>Specify the compression format of the input files. Choose one of the following options: GZIP, BZIP2, ZSTD, or No compression.</td>
</tr>
<tr>
<td>Delimiter character</td>
<td>Specify the ASCII character that separates fields in the input files. Nonprinting characters aren't supported.</td>
</tr>
<tr>
<td>NULL value as a string</td>
<td>Turn this option on if your data includes a null terminator. If this option is turned off, the Amazon Redshift COPY command treats null as an end of the record and ends the load process.</td>
</tr>
<tr>
<td>Sorting strategy</td>
<td>Use sorting to restart the extraction from the point of failure. Choose one of the following sorting strategies: Use sorting after the first fail (recommended), Use sorting if possible, or Never use sorting. For more information, see the section called “Sorting data” (p. 232).</td>
</tr>
<tr>
<td>Source temp schema</td>
<td>Enter the name of the schema in the source database, where the extraction agent can create the temporary objects.</td>
</tr>
<tr>
<td>Out file size (in MB)</td>
<td>Enter the size, in MB, of the files uploaded to Amazon S3.</td>
</tr>
<tr>
<td>Snowball out file size (in MB)</td>
<td>Enter the size, in MB, of the files uploaded to AWS Snowball. Files can be 1–1,000 MB in size.</td>
</tr>
<tr>
<td>Use table partitioning if it is supported by DB server and table size (in MB) is more than</td>
<td>Turn this option on to use table partitioning, and then enter the size of tables to partition.</td>
</tr>
<tr>
<td>Extract LOBs</td>
<td>Turn this option on to extract large objects (LOBs) from your source database. LOBs include BLOBs, CLOBs, NCLOBs, XML files, and so on. For every LOB, AWS SCT extraction agents create a data file.</td>
</tr>
<tr>
<td>Amazon S3 bucket LOBs folder</td>
<td>Enter the location for AWS SCT extraction agents to store LOBs.</td>
</tr>
<tr>
<td>Apply RTRIM to string columns</td>
<td>Turn this option on to trim a specified set of characters from the end of the extracted strings.</td>
</tr>
<tr>
<td>Keep files locally after upload to Amazon S3</td>
<td>Turn this option on to keep files on your local machine after data extraction agents upload them to Amazon S3.</td>
</tr>
</tbody>
</table>

Use the instructions in the following table to provide the information for **Amazon S3 settings**.
Sorting data

For this parameter | Do this
---|---
Use proxy | Turn this option on to use a proxy server to upload data to Amazon S3. Then choose the data transfer protocol, enter the host name, port, user name, and password.

Endpoint type | Choose FIPS to use the Federal Information Processing Standard (FIPS) endpoint. Choose VPCE to use the virtual private cloud (VPC) endpoint. Then for VPC endpoint, enter the Domain Name System (DNS) of your VPC endpoint.

Keep files on Amazon S3 after copying to Amazon Redshift | Turn this option on to keep extracted files on Amazon S3 after copying these files to Amazon Redshift.

Use the instructions in the following table to provide the information for Copy settings.

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum error count</td>
<td>Enter the number of load errors. After the operation reaches this limit, the AWS SCT data extraction agents end the data load process. The default value is 0, which means that the AWS SCT data extraction agents continue the data load regardless of the failures.</td>
</tr>
<tr>
<td>Replace not valid UTF-8 characters</td>
<td>Turn this option on to replace not valid UTF-8 characters with the specified character and continue the data load operation.</td>
</tr>
<tr>
<td>Use blank as null value</td>
<td>Turn this option on to load blank fields that consist of white space characters as null.</td>
</tr>
<tr>
<td>Use empty as null value</td>
<td>Turn this option on to load empty CHAR and VARCHAR fields as null.</td>
</tr>
<tr>
<td>Truncate columns</td>
<td>Turn this option on to truncate data in columns to fit the data type specification.</td>
</tr>
<tr>
<td>Automatic compression</td>
<td>Turn this option on to apply compression encoding during a copy operation.</td>
</tr>
<tr>
<td>Automatic statistics refresh</td>
<td>Turn this option on to refresh the statistics at the end of a copy operation.</td>
</tr>
<tr>
<td>Check file before load</td>
<td>Turn this option on to validate data files before loading them to Amazon Redshift.</td>
</tr>
</tbody>
</table>

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.
   - **Use sorting after first fail (recommended)** – The extraction agent first tries to get the data without using DENSE_RANK. If the first attempt fails, the extraction agent rebuilds the query using DENSE_RANK and preserves its location in case of failure.
4. Set additional parameters as described following, and then choose Create to create your data extraction task.

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.

You can save data extraction tasks to your local disk and restore them to the same or another project by using export and import. To export a task, make sure that you have at least one extraction task created in a project. You can import a single extraction task or all of the tasks created in the project.

When you export an extraction task, AWS SCT creates a separate `.xml` file for that task. The `.xml` file stores that task's metadata information, such as task properties, description, and subtasks. The `.xml` file doesn't contain information about processing of an extraction task. Information like the following is recreated when the task is imported:

- Task progress
- Subtask and stage states
• Distribution of extracting agents by subtasks and stages
• Task and subtask IDs
• Task name

Exporting and importing an AWS SCT data extraction task

You can quickly save an existing task from one project and restore it in another project (or the same project) using AWS SCT export and import. Use the following procedure to export and import data extraction tasks.

To export and import a data extraction task

1. For View, choose Data Migration view. The Agents tab appears.
2. Choose the Tasks tab. Your tasks are listed in the grid that appears.
3. Choose the three vertically aligned dots (ellipsis icon) located at the lower right corner under the list of tasks.
4. Choose Export task from the pop-up menu.
5. Choose the folder where you want AWS SCT to place the task export .xml file.
   
AWS SCT creates the task export file with a file name format of TASK-DESCRIPTION_TASK-ID.xml.
6. Choose the three vertically aligned dots (ellipsis icon) at lower right under the list of tasks.
7. Choose Import task from the pop-up menu.
   
   You can import an extraction task to a project connected to the source database, and the project has at least one active registered extraction agent.
8. Select the .xml file for the extraction task you exported.
   
   AWS SCT gets the extraction task's parameters from the file, creates the task, and adds the task to the extracting agents.
9. Repeat these steps to export and import additional data extraction tasks.

At the end of this process, your export and import are complete and your data extraction tasks are ready for use.

Data extraction using an AWS Snowball Edge device

The process of using AWS SCT and AWS Snowball Edge 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 Edge device, then an intermediate action where AWS copies the data from the AWS Snowball Edge 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 Edge.
1. Create an AWS Snowball Edge job using the AWS Snowball console. For more information, see Creating an AWS Snowball Edge Job in the AWS Snowball Edge Developer Guide.
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 AWS SCT.
7. Run and monitor the Local & DMS Task in AWS SCT.
8. Run the AWS SCT task and monitor progress in AWS SCT.

Step-by-step procedures for migrating data using AWS SCT and AWS Snowball Edge

The following sections provide detailed information on the migration steps.

**Step 1: Create an AWS Snowball Edge job**

Create an AWS Snowball job by following the steps outlined in the section Creating an AWS Snowball Edge Job in the AWS Snowball Edge Developer Guide.

**Step 2: Unlock the AWS Snowball Edge device**

Run the commands that unlock and provide credentials to the Snowball Edge device from the machine where you installed the AWS DMS agent. This way, you can be sure that the AWS DMS agent call connects to the AWS Snowball Edge device. For more information about unlocking the AWS Snowball Edge device, see Unlocking the Snowball Edge.

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

```
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, create a new AWS SCT project.

**To create a new project in AWS SCT**

2. Enter a name for your project, which is stored locally on your computer.
3. Enter the location for your local project file.
4. Choose OK to create your AWS SCT project.
5. Choose Add source to add a new source database to your AWS SCT project.
6. Choose Add target to add a new target platform in your AWS SCT project.
7. Choose the source database schema in the left panel.
8. In the right panel, specify the target database platform for the selected source schema.
9. Choose Create mapping. This button becomes active after you choose the source database schema and the target database platform.
Step 4: Install the source database driver for the AWS DMS agent on the Linux computer

For the migration to succeed, the AWS 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 AWS 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 one isn't 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, use the following.

```
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.

Update the `site_arep_login.sh` with the following code.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/microsoft/msodbcsql/lib64/
```

Simba ODBC Driver

Install the Microsoft ODBC Driver.

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 directory is /opt/sap, update the site_arep_login.sh with the following.

```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/OCS-16_0/lib:$SYBASE_HOME/OCS-16_0/
    lib3p64:$SYBASE_HOME/OCS-16_0/lib3p
```

The /etc/odbcinst.ini should include the following 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
Debug = 0
CommLog = 1
UsageCount = 2
```
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 & AWS 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 Edge 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.

   You can't migrate individual tables using AWS DMS and Snowball Edge.
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>Enter a name for the task.</td>
</tr>
<tr>
<td>Agent</td>
<td>Choose DMS Agent.</td>
</tr>
<tr>
<td>Replication Instance</td>
<td>Choose the AWS DMS replication instance that you want to use.</td>
</tr>
<tr>
<td>Migration Type</td>
<td>Choose the type of migration you want:</td>
</tr>
<tr>
<td></td>
<td>Choose Migrate existing data to migrate the contents of the chosen schema. This process is called a full load in AWS DMS.</td>
</tr>
<tr>
<td></td>
<td>Choose 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.</td>
</tr>
<tr>
<td>Target table preparation mode</td>
<td>Choose the preparation mode you want to use:</td>
</tr>
<tr>
<td></td>
<td>Truncate – Tables are truncated without affecting table metadata.</td>
</tr>
<tr>
<td></td>
<td>Drop tables on target – The tables are dropped and new tables are created in their place.</td>
</tr>
<tr>
<td></td>
<td>Do nothing – Data and metadata of the target tables aren't changed.</td>
</tr>
<tr>
<td>IAM role</td>
<td>Choose the predefined AWS Identity and Access Management (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 Prerequisites when using Amazon S3 as a source for AWS DMS.</td>
</tr>
<tr>
<td>For this parameter</td>
<td>Do this</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Compression format</td>
<td>Choose whether to have uploaded files compressed:</td>
</tr>
<tr>
<td></td>
<td><strong>GZIP</strong> – Files are compressed before loading. This is the default.</td>
</tr>
<tr>
<td></td>
<td><strong>No Compression</strong> – Extracts are faster but take more space.</td>
</tr>
<tr>
<td>Logging</td>
<td>Choose <strong>Enable</strong> to have Amazon CloudWatch create logs for the migration. You incur charges for this service. For more information about CloudWatch, see <a href="https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitoring.html">How Amazon CloudWatch works</a> in the Amazon CloudWatch User Guide.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter a description of the task.</td>
</tr>
<tr>
<td>S3 Bucket</td>
<td>Enter the name of an Amazon S3 bucket configured for this AWS Snowball Edge job in the AWS Snowball Edge console</td>
</tr>
<tr>
<td>Use AWS Snowball Edge</td>
<td>Choose this check box to use AWS Snowball Edge. If this box is not checked, then data is uploaded directly to the Amazon S3 bucket.</td>
</tr>
<tr>
<td>Job Name</td>
<td>Choose the AWS Snowball Edge job name that you created.</td>
</tr>
<tr>
<td>AWS Snowball Edge IP</td>
<td>Enter the IP address of the AWS Snowball Edge appliance.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the port value for the AWS Snowball Edge appliance.</td>
</tr>
<tr>
<td>Local Amazon S3 Access key</td>
<td>Enter the local AWS Snowball Edge access key that you retrieved from the device.</td>
</tr>
<tr>
<td>Local Amazon S3 Secret key</td>
<td>Enter the local AWS Snowball Edge secret key that you retrieved from the device.</td>
</tr>
</tbody>
</table>

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

**Step 7: Running and monitoring the AWS 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 AWS 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 AWS 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.
Data extraction task output

After your migration tasks complete, your data is ready. Use the following information to determine how to proceed based on the migration mode you chose and the location of your data.

<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 <a href="#">Connecting to clusters from client tools and code</a>.</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 <a href="#">Loading data from Amazon S3</a> in the Amazon Redshift documentation.</td>
</tr>
</tbody>
</table>

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 Amazon S3 bucket as shown following.
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 numeric, 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.

Also, if you are using AWS SCT to convert data from Netezza to Amazon Redshift, you can use native Netezza partitioning to manage migrating large tables. For more information, see Using native Netezza partitioning (p. 247).

**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 \textit{WHERE} clause, and you provide the range of values for each partition. To specify a list of values for the partitioned column, use the \textit{Values} box. You can load value information by using a .csv file.

The RANGE partition type creates default partitions at both ends of the partition values. These default partitions catch any data that is less than or greater than the specified partition values.

For example, you can create multiple partitions based on a value range that 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 AWS SCT.
2. Choose \textit{Data Migration view (other)} mode.
3. Choose the table where you want to set up virtual partitioning. Open the context (right-click) menu for the table, and choose \textit{Add virtual partitioning}.
4. In the \textit{Add virtual partitioning} dialog box, enter the information as follows.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition type</td>
<td>Choose \textit{RANGE}. The dialog box UI changes depending on the type you choose.</td>
</tr>
<tr>
<td>Column name</td>
<td>Choose the column that you want to partition.</td>
</tr>
<tr>
<td>Column type</td>
<td>Choose the data type for the values in the column.</td>
</tr>
<tr>
<td>Values</td>
<td>Add new values by typing each value in the \textit{New Value} box, then choosing the plus sign to add the value.</td>
</tr>
<tr>
<td>Load from file</td>
<td>(Optional) Enter the name of a .csv file that contains the partition values.</td>
</tr>
</tbody>
</table>
5. Choose \textit{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. To specify a list of values for the partitioned column, use the Values box. 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.

You can use the LIST partition type to filter the source data if you want to exclude particular values from the migration. For example, suppose that you want to omit rows with LO_ORDERKEY = 4. In this case, don't include the value 4 in the list of partition values and make sure that Include other values isn't chosen.

To create a LIST virtual partition

1. Open AWS SCT.
2. Choose Data Migration view (other) 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>Partition type</td>
<td>Choose LIST. The dialog box UI changes depending on the type you choose.</td>
</tr>
<tr>
<td>Column name</td>
<td>Choose the column that you want to partition.</td>
</tr>
<tr>
<td>New value</td>
<td>Type a value here to add it to the set of partitioning values.</td>
</tr>
<tr>
<td>Include other values</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>Load from file</td>
<td>(Optional) Enter 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 is an automated way of generating RANGE partitions. With DATA AUTO SPLIT, you tell AWS SCT the partitioning attribute, where to start and end, and the size of the range between the values. Then AWS SCT calculates the partition values automatically.

DATA AUTO SPLIT automates a lot of the work that is involved with creating range partitions. The tradeoff between using this technique and range partitioning is how much control you need over the partition boundaries. The automatic split process always creates equal size (uniform) ranges. Range
partitioning enables you to vary the size of each range as needed for your particular data distribution. For example, you can use daily, weekly, biweekly, monthly, and so on.

| 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 AWS SCT.
2. Choose Data Migration view (other) 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>Partition type</td>
<td>Choose DATE AUTO SPLIT. The dialog box UI changes depending on the type you choose.</td>
</tr>
<tr>
<td>Column name</td>
<td>Choose the column that you want to partition.</td>
</tr>
<tr>
<td>Start date</td>
<td>Type a start date.</td>
</tr>
<tr>
<td>End date</td>
<td>Type an end date.</td>
</tr>
<tr>
<td>Interval</td>
<td>Enter the interval unit, and choose the value for that unit.</td>
</tr>
</tbody>
</table>

5. Choose OK.

Using native Netezza partitioning

To speed up data migration, you can enable extraction agents to use the physical distribution of tables on a Netezza server. For example, after you create a project, you might collect statistics on a schema and analyze the size of the tables selected for migration. For tables that exceed the size specified, AWS SCT triggers the native Netezza partitioning mechanism.

To use native Netezza partitioning

1. Open AWS SCT, and choose New project for File. The New project dialog box appears.
2. Create a new project and connect to the source and target servers.
3. Choose View, and then choose Main view.
4. In the left panel that displays the schema from your source database, choose a schema. Open the context (right-click) menu for the object, and choose Collect statistics.
5. Choose each table to be migrated, and analyze the size and number of rows.
6. For Current project settings, choose the Data migration tab. Choose Use table partitioning ... if table is more than, and enter a table size limit in MB (for example, 100).
7. Register the required number of agents. For more information, see Registering extraction agents with the AWS Schema Conversion Tool (p. 228).
8. Create a data extraction task for the selected tables. For more information, see Creating, running, and monitoring an AWS SCT data extraction task (p. 234).
Check if large tables are split into subtasks, and that each subtask matches the dataset that presents a part of the table located on one Netezza slice.

9. Start and monitor the migration process until migration of the selected tables has finished.

**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 Amazon S3 bucket into the migrated data stored in Amazon Redshift.

**To migrate LOBs to Amazon Redshift**

1. Open an AWS SCT project.
2. Connect to the source and target databases. Refresh metadata from the target database, and make sure that the converted tables exist there.
3. For **Actions**, choose **Create local task**.
4. For **Migration mode**, choose one of the following:
   - **Extract and upload** to extract your data, and upload your data to Amazon S3.
   - **Extract, upload and copy** to extract your data, upload your data to Amazon S3, and copy it into your Amazon Redshift data warehouse.
5. Choose **Amazon S3 settings**.
6. For **Amazon S3 bucket LOBs folder**, enter the name of the folder in an Amazon S3 bucket where you want the LOBs stored.
   
   If you use AWS service profile, this field is optional. AWS SCT can use the default settings from your profile. To use another Amazon S3 bucket, enter the path here.
7. Turn on the **Use proxy** option to use a proxy server to upload data to Amazon S3. Then choose the data transfer protocol, enter the host name, port, user name, and password.
8. For **Endpoint type**, choose **FIPS** to use the Federal Information Processing Standard (FIPS) endpoint. Choose **VPCE** to use the virtual private cloud (VPC) endpoint. Then for **VPC endpoint**, enter the Domain Name System (DNS) of your VPC endpoint.
9. Turn on the **Keep files on Amazon S3 after copying to Amazon Redshift** option to keep extracted files on Amazon S3 after copying these files to Amazon Redshift.
10. Choose **Create** to create the task.

**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>
</tbody>
</table>
### Issue | Troubleshooting Suggestions
--- | ---
Contention delays | Avoid having too many agents accessing your data warehouse at the same time.

An agent goes down temporarily | 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.

An agent goes down permanently | 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.

---

**Migrating data from Apache Cassandra to Amazon DynamoDB**

You can use an AWS SCT data extraction agent to extract data from Apache Cassandra and migrate it to Amazon DynamoDB. The agent runs on an Amazon EC2 instance, where it extracts data from Cassandra, writes it to the local file system, and uploads it to an Amazon S3 bucket. You can then use AWS SCT to copy the data to DynamoDB.

Amazon DynamoDB is a NoSQL database service. To store data in DynamoDB, you create database tables and then upload data to those tables. The AWS SCT extraction agent for Cassandra automates the process of creating DynamoDB tables that match their Cassandra counterparts, and then populating those DynamoDB tables with data from Cassandra.

The process of extracting data can add considerable overhead to a Cassandra cluster. For this reason, you don't run the extraction agent directly against your production data in Cassandra. To avoid interfering with production applications, AWS SCT helps you create a clone data center—a standalone copy of the Cassandra data that you want to migrate to DynamoDB. The agent can then read data from the clone and make it available to AWS SCT, without affecting your production applications.

When the data extraction agent runs, it reads data from the clone data center and writes it to an Amazon S3 bucket. AWS SCT then reads the data from Amazon S3 and writes it to DynamoDB.

The following diagram shows the supported scenario.
If you are new to Cassandra, be aware of the following important terminology:

- A **node** is a single computer (physical or virtual) running the Cassandra software.
- A **server** is a logical entity composed of up to 256 nodes.
- A **rack** represents one or more servers.
- A **data center** is a collection of racks.
- A **cluster** is a collection of data centers.

For more information, go to the [Wikipedia page](https://en.wikipedia.org/wiki/Cassandra) for Apache Cassandra.
Use the information in the following topics to learn how to migrate data from Apache Cassandra to DynamoDB:

Topics
- Prerequisites for migrating from Cassandra to DynamoDB (p. 251)
- Create a new AWS SCT project (p. 254)
- Create a clone data center (p. 255)
- Install, configure, and run the data extraction agent (p. 260)
- Migrate data from the clone data center to Amazon DynamoDB (p. 263)
- Post-migration activities (p. 267)

Prerequisites for migrating from Cassandra to DynamoDB

Before you begin, you will need to perform several pre-migration tasks, as described in this section.

Supported Cassandra versions

AWS SCT supports the following Apache Cassandra versions:

- 3.11.2
- 3.1.1
- 3.0
- 2.1.20

Other versions of Cassandra aren't supported.

Amazon S3 settings

When the AWS SCT data extraction agent runs, it reads data from your clone data center and writes it to an 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 Storing AWS service profiles in the AWS SCT (p. 34).

Amazon EC2 instance for clone data center

As part of the migration process, you'll need to create a clone of an existing Cassandra data center. This clone will run on an Amazon EC2 instance that you provision in advance. The instance will run a standalone Cassandra installation, for hosting your clone data center independently of your existing Cassandra data center.

The new Amazon EC2 instance must meet the following requirements:

- Operating system: either Ubuntu or CentOS.
- Must have Java JDK 8 installed. (Other versions are not supported.)

To launch a new instance, go to the Amazon EC2 Management Console at https://console.aws.amazon.com/ec2/.
Security settings

AWS SCT communicates with the data extraction agent using Secure Sockets Layer (SSL). To enable SSL, set up a trust store and key store:

1. Launch AWS SCT.
2. From the Settings menu, choose Global Settings.
3. Choose the Security tab as shown following.
4. 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.
Create a new AWS SCT project

After you have performed the steps in Prerequisites for migrating from Cassandra to DynamoDB (p. 251), you're ready to create a new AWS SCT project for your migration. Follow these steps:

To create a new AWS SCT project using Apache Cassandra as a source and DynamoDB as a target

1. In AWS SCT, choose Add source.
2. Choose Cassandra, then choose Next.

   The Add source dialog box appears.
3. Provide the Apache Cassandra 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>Connection name</td>
<td>Enter a name for your database. AWS SCT displays this name in the tree in the left panel.</td>
</tr>
<tr>
<td>Server name</td>
<td>Enter the Domain Name Service (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Enter the port used to connect to your source database server.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Enter the user name and password to connect to your source database server. AWS SCT uses the password to connect to your source database only when you choose to connect to your database in a project. To guard against exposing the password for your source database, AWS SCT doesn't store the password by default. 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>Choose 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 store: The trust store to use.</td>
</tr>
<tr>
<td></td>
<td>• Key store: The key store to use.</td>
</tr>
<tr>
<td>Store password</td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. By turning this option on, you can store the database password and connect quickly to the database without having to enter the password.</td>
</tr>
</tbody>
</table>

4. Choose Test Connection to verify that AWS SCT can connect to your source database.
5. Choose Connect to connect to your source database.
6. In AWS SCT, choose Add target.
7. Choose Amazon DynamoDB, then choose Next.
Create a clone data center

To avoid interfering with production applications that use your Cassandra cluster, AWS SCT will create a clone data center and copy your production data into it. The clone data center acts as a staging area, so that AWS SCT can perform further migration activities using the clone rather than your production data center.

To begin the cloning process, follow this procedure:

1. In the AWS SCT window, on the left-hand side (source), expand the Datacenters node and choose one of your existing Cassandra data centers.
2. From the Actions menu, choose Clone Datacenter for Extract.
3. Read the introductory text, and then choose Next to continue.
4. In the Clone Datacenter for Extract window, add the following information:

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private IP:SSH port</td>
<td>Enter the private IP address and SSH port for any of the nodes in your Cassandra cluster. For example: 172.28.37.102:22</td>
</tr>
<tr>
<td>Public IP:SSH port</td>
<td>Enter the public IP address and SSH port for the node. For example: 41.184.48.27:22</td>
</tr>
<tr>
<td>OS User</td>
<td>Type a valid user name for connecting to the node.</td>
</tr>
<tr>
<td>OS Password</td>
<td>Type the password associated with the user name.</td>
</tr>
</tbody>
</table>
Create a clone data center

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key path</td>
<td>If you have an SSH private key (.pem file) for this node, choose Browse to navigate to the location where the private key is stored.</td>
</tr>
<tr>
<td>Passphrase</td>
<td>If your SSH private key is protected by a passphrase, type it here.</td>
</tr>
<tr>
<td>JMX user</td>
<td>Enter the JMX user name for accessing your Cassandra cluster.</td>
</tr>
<tr>
<td>JMX password</td>
<td>Type the password associated with the JMX user.</td>
</tr>
</tbody>
</table>

Choose Next to continue. AWS SCT connects to the Cassandra node, where it runs the nodetool status command.

5. In the Source Cluster Parameters window, accept the default values, and choose Next to continue.

6. In the Node Parameters window, verify the connection details for all of the nodes in the source cluster. AWS SCT will fill in some of these details automatically; however, you must supply any missing information.

   **Note**
   Instead of entering all of the data here, you can bulk-upload it instead. To do this, choose Export to create a .csv file. You can then edit this file, adding a new line for each node in your cluster. When you are done, choose Upload. AWS SCT will read the .csv file and use it to populate the Node parameters window.

Choose Next to continue. AWS SCT verifies that the node configuration is valid.

7. In the Configure Target Datacenter window, review the default values. In particular, note the Datacenter suffix field: When AWS SCT creates your clone data center, it will be named similarly to the source data center, but with the suffix that you provide. For example, if the source data center is named my_datacenter, then a suffix of _tgt would cause the clone to be named my_datacenter_tgt.

8. While still in the Configure Target Datacenter window, choose Add new node:
Step 1. Introduction

Step 2. Configure source datacenter
2.1. Source cluster parameters
2.2. Node parameters

Step 3. Configure target datacenter

Step 4. Datacenter Synchronization

Step 5. Summary

Before starting any modification, please read the policy for using the current data center. Also, use LOCAL_QUORUM/LOCAL

SCT will now verify network compatibility between the source and target datacenters. Provide the parameters for the target datacenter verification process. You can also click “Show Log” to view the logs.

Target Cassandra datacenter

Hints directory: /var/lib/cassandra/hints
Data file directory: /var/lib/cassandra/data
Commitlog directory: /var/lib/cassandra/commitlog
CDC raw directory: /var/lib/cassandra/cdc
Snitch mode: Ec2Snitch
datacenter suffix: _tgt

Specify target node. Add new node or Select...

Show Cassandra configuration

Show Log
9. In the **Add New Node** window, add the information needed to connect to the Amazon EC2 instance that you created in *Amazon EC2 instance for clone data center* (p. 251).

When the settings are as you want them, choose **Add**. The node appears in the list:
10. Choose **Next** to continue. The following confirmation box appears:

![Confirmation](image)

Choose **OK** to continue. AWS SCT reboots your source data center, one node at a time.

11. Review the information in the **Datacenter Synchronization** window. If your cluster is running Cassandra version 2, then AWS SCT copies all of the data to the clone data center. If your cluster is running Cassandra version 3, then you can choose which keyspace or keyspaces you want to copy to the clone data center.

12. When you are ready to begin replicating data to your clone data center, choose **Start**.

Data replication will begin immediately. AWS SCT displays a progress bar so that you can monitor the replication process. Note that replication can take a long time, depending on how much data is in the source data center. If you need to cancel the operation before it's fully complete, choose **Cancel**.

When the replication is complete, choose **Next** to continue.

13. In the Summary window, AWS SCT displays a report showing the state of your Cassandra cluster, along with next steps.

Review the information in this report, and then choose **Finish** to complete the wizard.

---

### Install, configure, and run the data extraction agent

Now that you have a clone of your data center, you are ready to begin using the AWS SCT data extraction agent for Cassandra. This agent is available as part of the AWS SCT distribution (for more information, see [Installing, verifying, and updating AWS SCT](p. 5)).

**Note**

We recommend that you run the agent on an Amazon EC2 instance. The Amazon EC2 instance must meet the following requirements:

- Operating system: either Ubuntu or CentOS.
- 8 virtual CPUs, at a minimum.
- At least 16GB of RAM.

If you don't already have an Amazon EC2 instance that meets these requirements, go to the Amazon EC2 Management Console ([https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/)) and launch a new instance before proceeding.

Follow this procedure to install, configure, and run the AWS SCT data extraction agent for Cassandra:

1. Log in to your Amazon EC2 instance.
2. Verify that you are running Java 1.8.x is installed:

```java
java -version
```

3. Install the sshfs package:

```bash
sudo yum install sshfs
```

4. Install the expect package:

```bash
sudo yum install expect
```

5. Edit the `/etc/fuse.conf` file, and uncomment the string `user_allow_other`:

```yaml
# mount_max = 1000
user_allow_other
```

6. The AWS SCT data extraction agent for Cassandra is available as part of the AWS SCT distribution (for more information, see Installing, verifying, and updating AWS SCT (p. 5)). You can find the agent in the .zip file that contains the AWS SCT installer file, in the `agents` directory. The following builds of the agent are available.

<table>
<thead>
<tr>
<th>File name</th>
<th>Operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws-cassandra-extractor-n.n.n.deb</td>
<td>Ubuntu</td>
</tr>
<tr>
<td>aws-cassandra-extractor-n.n.n.x86_64.rpm</td>
<td>CentOS</td>
</tr>
</tbody>
</table>

Choose the file that's appropriate for your Amazon EC2 instance. Use the `scp` utility to upload that file to your Amazon EC2 instance.

7. Install the AWS SCT data extraction agent for Cassandra. (Replace `n.n.n` with the build number.)

   - For Ubuntu:
     ```bash
     sudo dpkg -i aws-cassandra-extractor-n.n.n.deb
     ```

   - For CentOS:
     ```bash
     sudo yum install aws-cassandra-extractor-n.n.n.x86_64.rpm
     ```

   During the installation process, you'll be asked to select the Cassandra version you want to work with. Choose version 3 or 2, as appropriate.

8. After the installation completes, review the following directories to ensure that they were created successfully:

   - `/var/log/cassandra-data-extractor/`—for extraction agent logs.
   - `/etc/cassandra-data-extractor/`—for the agent configuration file (`agent-settings.yaml`).

9. To enable the agent to communicate with AWS SCT, you must have a key store and a trust store available. (You created these in Security settings (p. 252).) Use the `scp` utility to upload these files to your Amazon EC2 instance.
The configuration utility (see next step) requires you to specify the key store and trust store, so you need to have them available.

10. Run the configuration utility:

```
sudo java -jar /usr/share/aws-cassandra-extractor/aws-cassandra-extractor.jar --configure
```

The utility will prompt you for several configuration values. You can use the following example as a guide, while substituting your own values:

```
Enter the number of data providers nodes [1]: 1
Enter IP for Cassandra node 1: 34.220.73.140
Enter SSH port for Cassandra node <34.220.73.140> [22]: 22
Enter SSH login for Cassandra node <34.220.73.140> : centos
Enter SSH password for Cassandra node <34.220.73.140> (optional):
Is the connection to the node using a SSH private key? Y/N [N] : Y
Enter the path to the private SSH key for Cassandra node <34.220.73.140>: /home/centos/my-ec2-private-key.pem
Enter passphrase for SSH private key for Cassandra node <34.220.73.140> (optional):
Enter the path to the cassandra.yaml file location on the node <34.220.73.140>: /etc/cassandra/conf/
Enter the path to the Cassandra data directories on the node <34.220.73.140>: /u01/cassandra/data
===== Mounting process started =====
Node [34.220.73.140] mounting started.
Will be executed command:
sudo sshfs ubuntu@34.220.73.140:/etc/cassandra/ /mnt/aws-cassandra-data-extractor/34.220.73.140_node/conf/ -p 22 -o allow_other -o StrictHostKeyChecking=no -o IdentityFile=/home/ubuntu/dbbest-ec2-oregon_s.pem > /var/log/aws-cassandra-data-extractor/dmt-cassandra-v3/conf_34.220.73.140.log 2>&1
Will be executed command:
sudo sshfs ubuntu@34.220.73.140:/u01/cassandra/data/ /mnt/aws-cassandra-data-extractor/34.220.73.140_node/data/data -p 22 -o allow_other -o StrictHostKeyChecking=no -o IdentityFile=/home/ubuntu/dbbest-ec2-oregon_s.pem > /var/log/aws-cassandra-data-extractor/dmt-cassandra-v3/data_34.220.73.140.log 2>&1
===== Mounting process was over =====
Enable SSL communication Y/N [N] : Y
Path to key store: /home/centos/Cassandra_key
Key store password:123456
Re-enter the key store password:123456
Path to trust store: /home/centos/Cassandra_trust
Trust store password:123456
Re-enter the trust store password:123456
Enter the path to the output local folder: /home/centos/out_data
=== Configuration aws-agent-settings.yaml successful completed ===
```

**Note**

When the configuration utility has completed, you might see the following message:

Change the SSH private keys permission to 600 to secure them. You can also set permissions to 400.

You can use the `chmod` command to change the permissions, as in this example:

```
chmod 400 /home/centos/my-ec2-private-key.pem
```

11. After the configuration utility completes, review the following directories and files:

- `/etc/cassandra-data-extractor/agent-settings.yaml`—the settings file for the agent.
- `$HOME/out_data`—a directory for extraction output files.
- `/mnt/cassandra-data-extractor/34.220.73.140_node/conf`—an empty Cassandra home folder. (Replace `34.220.73.140` with your actual IP address.)
- `/mnt/cassandra-data-extractor/34.220.73.140_node/data/data`—an empty Cassandra data file. (Replace `34.220.73.140` with your actual IP address.)

If these directories aren't mounted, use the following command to mount them:

```
sudo java -jar /usr/share/aws-cassandra-extractor/aws-cassandra-extractor.jar -mnt
```

12. Mount the Cassandra home and data directories:

```
sudo java -jusr/share/cassandra-extractor/rest-extraction-service.jar -mnt
```

After the mounting process is complete, review the Cassandra home folder and Cassandra data file directory as shown in the following example. (Replace `34.220.73.140` with your actual IP address.)

```
ls -l /mnt/cassandra-data-extractor/34.220.73.140_node/conf
ls -l /mnt/cassandra-data-extractor/34.220.73.140_node/data/data
```

13. Start the AWS SCT data extraction agent for Cassandra:

```
sudo systemctl start aws-cassandra-extractor
```

**Note**

By default, the agent runs on port 8080. You can change this by editing the `agent-settings.yaml` file.

---

**Migrate data from the clone data center to Amazon DynamoDB**

You are now ready to perform the migration from the clone data center to Amazon DynamoDB, using AWS SCT. AWS SCT manages the workflows among the AWS SCT data extraction agent for Cassandra, AWS Database Migration Service (AWS DMS), and DynamoDB. You perform the migration process entirely within the AWS SCT interface, and AWS SCT manages all of the external components on your behalf.

To migrate your data, follow this procedure:

**To migrate your data from the clone data center to DynamoDB**

1. From the View menu, choose Data migration view.
2. Choose the Agents tab.
3. If you haven't yet registered the AWS SCT data extraction agent, you'll see the following message:
Choose Register.

4. In the **New agent registration** window, add the following information:

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Type a short description for this agent.</td>
</tr>
<tr>
<td><strong>Host name</strong></td>
<td>Enter the hostname of the Amazon EC2 instance you used for Install, configure, and run the data extraction agent (p. 260)</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Enter the port number for the agent. (The default port number is 8080.)</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>If you are using SSL, leave this field blank; otherwise, type the password for logging into the host.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>If you are using SSL, choose this option to activate the SSL tab.</td>
</tr>
</tbody>
</table>

If you are using SSL, choose the **SSL** tab and add the following information:

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trust store</strong></td>
<td>Choose the trust store you configured in Install, configure, and run the data extraction agent (p. 260).</td>
</tr>
<tr>
<td><strong>Key store</strong></td>
<td>Choose the key store you configured in Install, configure, and run the data extraction agent (p. 260).</td>
</tr>
</tbody>
</table>

When the settings are as you want them, choose Register. AWS SCT will attempt to connect with the AWS SCT data extraction agent for Cassandra.

5. On the left side of the AWS SCT window, choose the Cassandra data center that you created in Create a clone data center (p. 255).

6. From the **Actions** menu, choose **Create Local & DMS Task**.

7. In the **Create Local & DMS Task** window, enter the following information:

<table>
<thead>
<tr>
<th>For this parameter</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task name</strong></td>
<td>Type a short name for the AWS DMS task to be created.</td>
</tr>
<tr>
<td><strong>Replication instance</strong></td>
<td>Choose the AWS DMS replication instance that you want to use.</td>
</tr>
<tr>
<td><strong>Migration type</strong></td>
<td>Choose <strong>Migrate existing data and replication ongoing changes</strong>. This will migrate the tables in your Cassandra clone data center to DynamoDB, and then capture all ongoing changes. This process is called full load and CDC in AWS DMS.</td>
</tr>
<tr>
<td><strong>Target table preparation mode</strong></td>
<td>If you already have corresponding tables in DynamoDB and want to delete them prior to migration, choose</td>
</tr>
</tbody>
</table>
For this parameter | Do this
--- | ---
Drop tables on target. Otherwise, leave the setting at its default value (Do nothing).
IAM role | Choose the predefined IAM role that has permissions to access the Amazon S3 bucket and the target database (Amazon DynamoDB). For more information about the permissions required to access an Amazon S3 bucket, see Amazon S3 settings (p. 251).
Logging level | Choose an appropriate logging level for the migration task.
Description | Type a description for the task.
Data encryption | Choose either Enable or Disable.
Delete files from the local directory | Choose this option to delete data files from the agent's local directory after it loads the files to Amazon S3.
S3 bucket | Enter the name of an Amazon S3 bucket for which you have write privileges.

When the settings are as you want them, choose Create.

8. Choose the Tasks tab, where you should see the task you created. To start the task choose Start.

You can monitor the task progress, as shown in the following screenshot:
### AWS Schema Conversion Tool User Guide

**Migrate data from the clone data center to Amazon DynamoDB**

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Status</th>
<th>Full load progress %</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws-dms-cassandra-v2-perf-test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aws-dms-cassandra-v2-perf-test(Extract)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aws-dms-cassandra-v2-perf-test(Upload)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aws-dms-cassandra-v2-perf-test(DMS)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Task details**

- **ID:** 581a237c99c84e8f9fb887f4baf43671
- **Task name:** aws-dms-cassandra-v2-perf-test
- **Status:** STOPPED
- **Migration type:** full-load-and-cdc
- **Description:** aws-dms-cassandra-v2-perf-test
- **Number of nodes:** 1
- **Bucket name:** awscsvfile
- **Logging level:** INFO
- **Extractors log path:** 
- **Last failure message:**

---

**Download log**

- **Resume**
- **Stop**
- **Delete**

---

**Version 1.0**

266
Post-migration activities

If you are finished with the migration and want to delete the migration task, do the following:

1. Choose the **Tasks** tab.
2. If your task is currently running, choose **Stop**.
3. To delete the task, choose **Delete**.

If you no longer need to use the AWS SCT data extraction agent for Cassandra, do the following:

1. Choose the **Agents** tab.
2. Choose the agent you no longer need.
3. Choose **Unregister**.
Converting application SQL using AWS SCT

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. Also, for an Oracle to PostgreSQL conversion, you can use AWS SCT to convert SQL code embedded into C#, C++, Java, and Pro*C applications.

Topics

• Overview of converting application SQL (p. 268)
• Converting SQL code in your applications with AWS SCT (p. 268)
• Converting SQL code in C# applications with AWS SCT (p. 275)
• Converting SQL code in C++ applications with AWS SCT (p. 279)
• Converting SQL code in Java applications with AWS SCT (p. 285)
• Converting SQL code in Pro*C applications with AWS SCT (p. 290)

Overview of converting application SQL

To convert the SQL code in your application, 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 generic application conversion projects in AWS SCT (p. 269).

• 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 in AWS SCT (p. 272).

• 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 AWS SCT assessment report in AWS SCT (p. 273).

• 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 AWS SCT (p. 274).

Converting SQL code in your applications with AWS SCT

You can use AWS SCT to convert SQL code embedded into your applications. The generic AWS SCT application converter treats your application code as plain text. It scans your application code and
extracts SQL code with regular expressions. This converter supports different types of source code files and works with application code that is written in any programming language.

The generic application converter has the following limitations. It doesn't dive deep into the application logic that is specific for the programming language of your application. Also, the generic converter doesn't support SQL statements from different application objects, such as functions, parameters, local variables, and so on.

To improve your application SQL code conversion, use language-specific application SQL code converters. For more information, see Converting SQL code in C# applications (p. 275), Converting SQL code in Java applications (p. 285), and Converting SQL code in Pro*C applications (p. 290).

Creating generic application conversion projects in AWS SCT

In the AWS Schema Conversion Tool, 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 a generic application conversion project.

To create an application conversion project

1. In the AWS Schema Conversion Tool, choose New generic application on 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><strong>Name</strong></td>
<td>Enter 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><strong>Location</strong></td>
<td>Enter the location of the source code for your application.</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>Choose one of the following:</td>
</tr>
<tr>
<td></td>
<td>• Java</td>
</tr>
<tr>
<td></td>
<td>• C++</td>
</tr>
<tr>
<td></td>
<td>• C#</td>
</tr>
<tr>
<td></td>
<td>• Any</td>
</tr>
<tr>
<td><strong>Target parameter style</strong></td>
<td>Choose the syntax to use for bind variables in the converted code. Different database platforms use different syntax for bind variables. Choose one of the following:</td>
</tr>
<tr>
<td></td>
<td>• Same as in source</td>
</tr>
</tbody>
</table>
For this parameter | Do this
--- | ---
• Positional (?)
• Indexed (:1)
• Indexed ($1)
• Named (@name)
• Named (:name)
• Named (&name)
• Named ($name)
• Named (#name)
• Named (!name!)

Choose the source database schema | In the source tree, choose the schema that your application uses. Make sure that this schema is part of a mapping rule.
--- | ---

3. Select **Don't cast bind variables to SQL types** to avoid conversion of bind variables types to SQL types. This option is available only for an Oracle to PostgreSQL conversion.

For example, your source application code includes the following Oracle query:

```sql
SELECT * FROM ACCOUNT WHERE id = ?
```

When you select **Don't cast bind variables to SQL types**, AWS SCT converts this query as shown following.

```sql
SELECT * FROM account WHERE id = ?
```

When you clear **Don't cast bind variables to SQL types**, AWS SCT changes the bind variable type to the **NUMERIC** data type. The conversion result is shown following.

```sql
SELECT * FROM account WHERE id = (?)::NUMERIC
```

4. Select **Keep object names** to avoid adding the schema name to the name of the converted object. This option is available only for an Oracle to PostgreSQL conversion.

For example, suppose that your source application code includes the following Oracle query.

```sql
SELECT * FROM ACCOUNT
```

When you select **Keep object names**, AWS SCT converts this query as shown following.

```sql
SELECT * FROM account
```

When you clear **Keep object names**, AWS SCT adds the schema name to the name of the table. The conversion result is shown following.

```sql
SELECT * FROM schema_name.account
```

If your source code includes the names of the parent objects in the names of the objects, AWS SCT uses this format in the converted code. In this case, ignore the **Keep object names** option because AWS SCT adds the names of the parent objects in the converted code.

5. Choose **OK** to create your application conversion project.
Managing application conversion projects in AWS SCT

You can open an existing application conversion project and add multiple application conversion projects.

After you create an application conversion project, the project window opens automatically. You can close the application conversion project window and get back to it later.

To open an existing application conversion project

1. In the left panel, choose the application conversion project node, and open the context (right-click) menu.
2. Choose Manage application.

To add an additional application conversion project

1. In the left panel, choose the application conversion project node, and open the context (right-click) menu.
2. Choose New application.
3. Enter the information that is required to create a new application conversion project. For more information, see Creating generic application conversion projects (p. 269).

Analyzing and converting your SQL code in AWS SCT

Use the following procedure to analyze and convert your SQL code in the AWS Schema Conversion Tool.

To analyze and convert your SQL code

1. Open an existing application conversion project, and choose Analyze.
AWS SCT analyzes your application code and extracts the SQL code. AWS SCT displays the extracted SQL code in the **Parsed SQL scripts** list.

2. For **Parsed SQL scripts**, choose an item to review its extracted SQL code. AWS SCT displays the code of the selected item in the **Extracted SQL script** pane.

3. Choose **Convert** to convert the SQL code the **Extracted SQL script** pane. AWS SCT converts the code to a format compatible with your target database.

You can edit the converted SQL code. For more information, see Editing and saving your converted SQL code (p. 274).

4. When you create an application conversion assessment report, AWS SCT converts all extracted SQL code items. For more information, see Creating and using the assessment report (p. 273).

---

**Creating and using the AWS SCT assessment report in AWS SCT**

The *application conversion assessment report* provides information about converting the application SQL code to a format compatible with your target database. The report details all extracted SQL code, all converted SQL code, and action items for SQL code that AWS SCT can't convert.

**Creating an application conversion assessment report**

Use the following procedure to create an application conversion assessment report.

**To create an application conversion assessment report**

1. In the application conversion project window, choose **Create report** on the **Actions** menu.
AWS SCT creates the application conversion assessment report and opens it 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.

   ![Application Assessment Report](image)

3. Choose **SQL extraction actions**.

   Review the list of SQL code items that AWS SCT can't extract from your source code.

4. Choose **SQL conversion actions**.

   Review the list of SQL code items that AWS SCT can't convert automatically. Use recommended actions to manually convert the SQL code. For information about how to edit your converted SQL code, see Editing and saving your converted SQL code with AWS SCT (p. 274).

5. (Optional) Save a local copy of the report as either a PDF file or a comma-separated values (CSV) file:
   - Choose **Save to PDF** at upper right to save the report as a PDF file.
     
     The PDF file contains the executive summary, action items, and recommendations for application conversion.
   - Choose **Save to CSV** at upper right to save the report as a CSV file.
     
     The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the SQL code.

**Editing and saving your converted SQL code with AWS SCT**

The assessment report includes a list of SQL code items that AWS SCT can't convert. For each item, AWS SCT creates 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.

   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.
Converting SQL code in C# applications with AWS SCT

For an Oracle to PostgreSQL conversion, you can use AWS Schema Conversion Tool (AWS SCT) to convert SQL code embedded into your C# applications. This specific C# application converter understands the application logic. It collects statements that are located in different application objects, such as functions, parameters, local variables, and so on.

Because of this deep analysis, the C# application SQL code converter provides better conversion results than the generic converter.

Creating C# application conversion projects in AWS SCT

You can create a C# application conversion project only for converting Oracle database schemas to PostgreSQL database schemas. Make sure that you add a mapping rule in your project that includes a source Oracle schema and a target PostgreSQL database. For more information, see Creating mapping rules in AWS SCT (p. 122).

You can add multiple application conversion projects in a single AWS SCT project. Use the following procedure to create a C# application conversion project.

To create a C# application conversion project

1. Create a database conversion project, and add a source Oracle database. For more information, see Creating an AWS SCT project (p. 16) and Adding database servers to an AWS SCT project (p. 17).
2. Add a mapping rule that includes your source Oracle database and a target PostgreSQL database. You can add a target PostgreSQL database or use a virtual PostgreSQL target database platform in a mapping rule. For more information, see Creating mapping rules in AWS SCT (p. 122) and Using virtual targets (p. 123).
3. On the View menu, choose Main view.
4. On the Applications menu, choose New C# application.

The Creating a C# application conversion project dialog box appears.
5. For Name, enter a name for your C# application conversion project. Because each database schema conversion project can have one or more child application conversion projects, choose a name that makes sense if you add multiple projects.

6. For Location, enter the location of the source code for your application.

7. In the source tree, choose the schema that your application uses. Make sure that this schema is part of a mapping rule. AWS SCT highlights the schemas that are part of a mapping rule in bold.

8. Choose OK to create your C# application conversion project.

9. Find your C# application conversion project in the Applications node in the left panel.

Managing C# application conversion projects in AWS SCT

You can add multiple C# application conversion projects, update the application code in the AWS SCT project, or remove a C# conversion project from your AWS SCT project.

To add an additional C# application conversion project

1. Expand the Applications node in the left panel.
2. Choose the C# node, and open the context (right-click) menu.
3. Choose New application.
4. Enter the information that is required to create a new C# application conversion project. For more information, see Creating C# application conversion projects (p. 275).

After you make changes in your source application code, upload it into the AWS SCT project.

To upload the updated application code

1. Expand the C# node under Applications in the left panel.
2. Choose the application to update, and open the context (right-click) menu.
3. Choose **Refresh** and then choose **Yes**.

AWS SCT uploads your application code from the source files and removes conversion results. To keep code changes that you made in AWS SCT and the conversion results, create a new C# conversion project.

**To remove a C# application conversion project**

1. Expand the C# node under Applications in the left panel.
2. Choose the application to remove, and open the context (right-click) menu.
3. Choose **Delete** and then choose **OK**.

**Converting your C# application SQL code in AWS SCT**

After you add your C# application to the AWS SCT project, convert SQL code from this application to a format compatible with your target database platform. Use the following procedure to analyze and convert the SQL code embedded in your C# application in the AWS Schema Conversion Tool.

**To convert your SQL code**

1. Expand the C# node under Applications in the left panel.
2. Choose the application to convert, and open the context (right-click) menu.
3. Choose **Convert**. AWS SCT analyzes your source code files, determines the application logic, and loads code metadata into the project. This code metadata includes C# classes, objects, methods, global variables, interfaces, and so on.

In the target database panel, AWS SCT creates the similar folders structure to your source application project. Here you can review the converted application code.
4. Save your converted application code. For more information, see Saving your converted application code (p. 279).

Creating a C# application conversion assessment report in AWS SCT

The C# application conversion assessment report provides information about converting the SQL code embedded in your C# application to a format compatible with your target database. The assessment report provides conversion details for all SQL execution points and all source code files. The assessment report also includes action items for SQL code that AWS SCT can't convert.

Use the following procedure to create a C# application conversion assessment report.

To create a C# application conversion assessment report

1. Expand the C# node under Applications in the left panel.
2. Choose the application to convert, and open the context (right-click) menu.
3. Choose Convert.
4. On the View menu, choose Assessment report view.
5. View the Summary tab.

The Summary tab, shown following, displays the executive summary information from the C# application assessment report. It shows conversion results for all SQL execution points and all source code files.

6. Choose Save statements to JSON to save the extracted SQL code from your C# application as a JSON file.
7. (Optional) Save a local copy of the report as either a PDF file or a comma-separated values (CSV) file:
   - Choose Save to PDF at upper right to save the report as a PDF file.
     The PDF file contains the executive summary, action items, and recommendations for application conversion.
   - Choose Save to CSV at upper right to save the report as a CSV file.
     The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the SQL code.
Saving your converted application code with AWS SCT

Use the following procedure to save your converted application code.

To save your converted application code
1. Expand the C# node under Applications in the target database panel.
2. Choose your converted application, and choose Save.
3. Enter the path to the folder to save the converted application code, and choose Select folder.

Converting SQL code in C++ applications with AWS SCT

For an Oracle to PostgreSQL conversion, you can use AWS SCT to convert SQL code embedded into your C++ applications. This specific C++ application converter understands the application logic. It collects statements that are located in different application objects, such as functions, parameters, local variables, and so on.

Because of this deep analysis, the C++ application SQL code converter provides better conversion results than the generic converter.

Creating C++ application conversion projects in AWS SCT

You can create a C++ application conversion project only for converting Oracle database schemas to PostgreSQL database schemas. Make sure that you add a mapping rule in your project that includes a source Oracle schema and a target PostgreSQL database. For more information, see Creating mapping rules in AWS SCT (p. 122).

You can add multiple application conversion projects in a single AWS SCT project.

To create a C++ application conversion project
1. Create a database conversion project, and add a source Oracle database. For more information, see Creating an AWS SCT project (p. 16) and Adding database servers to an AWS SCT project (p. 17).
2. Add a mapping rule that includes your source Oracle database and a target PostgreSQL database. You can add a target PostgreSQL database or use a virtual PostgreSQL target database platform in a mapping rule. For more information, see Creating mapping rules in AWS SCT (p. 122) and Using virtual targets (p. 123).
3. On the View menu, choose Main view.
4. On the Applications menu, choose New C++ application.

The Creating a C++ application conversion project dialog box appears.
5. For **Name**, enter a name for your C++ application conversion project. Because each database schema conversion project can have one or more child application conversion projects, choose a name that makes sense if you add multiple projects.

6. For **Location**, enter the location of the source code for your application.

7. In the source tree, choose the schema that your application uses. Make sure that this schema is part of a mapping rule. AWS SCT highlights the schemas that are part of a mapping rule in bold.

8. Choose **OK** to create your C++ application conversion project.

9. Find your C++ application conversion project in the **Applications** node in the left panel.

### Converting your C++ application SQL code in AWS SCT

After you add your C++ application to the AWS SCT project, convert SQL code from this application to a format compatible with your target database platform. Use the following procedure to analyze and convert the SQL code embedded in your C++ application in AWS SCT.

**To convert your SQL code**

1. Expand the **C++** node under **Applications** in the left panel, and choose the application to convert.

2. In the **Source Oracle application project**, choose **Settings**. Review and edit the conversion settings for the selected C++ application. You can also specify the conversion settings for all C++ applications that you added to your AWS SCT project. For more information, see *Managing C++ application conversion projects* (p. 283).
3. For **Compiler type**, choose the compiler that you use for the source code of your C++ application. AWS SCT supports the following C++ compilers: Microsoft Visual C++, GCC, the GNU Compiler Collection, and Clang. The default option is Microsoft Visual C++.

4. For **User-defined macros**, enter the path to the file that includes user-defined macros from your C++ project. Make sure that this file has the following structure: `#define name value`. In the preceding example, `value` is an optional parameter. The default value for this optional parameter is 1.

   To create this file, open your project in Microsoft Visual Studio, and then choose **Project, Properties, C/C++, and Preprocessor**. For **Preprocessor definitions**, choose **Edit** and copy names and values to a new text file. Then, for each string in the file, add the following prefix: `#define`.

5. For **External include directories**, enter the paths to the folders that include external libraries that you use in your C++ project.

6. In the left pane, choose the application to convert, and open the context (right-click) menu.

7. Choose **Convert**. AWS SCT analyzes your source code files, determines the application logic, and loads code metadata into the project. This code metadata includes C++ classes, objects, methods, global variables, interfaces, and so on.

   In the target database panel, AWS SCT creates the similar folders structure to your source application project. Here you can review the converted application code, as shown following.
Managing C++ application conversion projects in AWS SCT

You can add multiple C++ application conversion projects, edit conversion settings, update the C++ application code, or remove a C++ conversion project from your AWS SCT project.

To add an additional C++ application conversion project

1. Expand the Applications node in the left panel.
2. Choose the C++ node, and open the context (right-click) menu.
3. Choose New application.
4. Enter the information that is required to create a new C++ application conversion project. For more information, see Creating C++ application conversion projects (p. 279).

You can specify conversion settings for all C++ application conversion projects in your AWS SCT project.

To edit conversion settings for all C++ applications

1. On the Settings menu, choose Project settings, and then choose Application conversion.
2. For Compiler type, choose the compiler that you use for the source code of your C++ application. AWS SCT supports the following C++ compilers: Microsoft Visual C++, GCC, the GNU Compiler Collection, and Clang. The default option is Microsoft Visual C++.
3. For User-defined macros, enter the path to the file that includes user-defined macros from your C++ project. Make sure that this file has the following structure: `#define name value`. In the preceding example, `value` is an optional parameter. The default value for this optional parameter is `1`.
   
   To create this file, open your project in Microsoft Visual Studio, and then choose Project, Properties, C/C++, and Preprocessor. For Preprocessor definitions, choose Edit and copy names and values to a new text file. Then, for each string in the file, add the following prefix: `#define`.
4. For External include directories, enter the paths to the folders that include external libraries that you use in your C++ project.
5. Choose OK to save the project settings and close the window.

Or you can specify conversion settings for each C++ application conversion project. For more information, see Converting your C++ application SQL code (p. 280).

After you make changes in your source application code, upload it into the AWS SCT project.

To upload the updated application code

1. Expand the C++ node under Applications in the left panel.
2. Choose the application to update, and open the context (right-click) menu.
3. Choose Refresh and then choose Yes.

AWS SCT uploads your application code from the source files and removes conversion results. To keep code changes that you made in AWS SCT and the conversion results, create a new C++ conversion project.
Also, AWS SCT removes the application conversion settings that you specified for the selected application. After you upload the updated application code, AWS SCT applies the default values from the project settings.

To remove a C++ application conversion project
1. Expand the C++ node under Applications in the left panel.
2. Choose the application to remove, and open the context (right-click) menu.
3. Choose Delete and then choose OK.

Creating a C++ application conversion assessment report in AWS SCT

The C++ application conversion assessment report provides information about converting the SQL code embedded in your C++ application to a format compatible with your target database. The assessment report provides conversion details for all SQL execution points and all source code files. The assessment report also includes action items for SQL code that AWS SCT can’t convert.

To create a C++ application conversion assessment report
1. Expand the C++ node under Applications in the left panel.
2. Choose the application to convert, and open the context (right-click) menu.
3. Choose Convert.
4. On the View menu, choose Assessment report view.
5. View the Summary tab.

The Summary tab displays the executive summary information from the C++ application assessment report. It shows conversion results for all SQL execution points and all source code files.
6. Choose Save statements to JSON to save the extracted SQL code from your Java application as a JSON file.
7. (Optional) Save a local copy of the report as either a PDF file or a comma-separated values (CSV) file:
   - Choose Save to PDF at upper right to save the report as a PDF file.
     The PDF file contains the executive summary, action items, and recommendations for application conversion.
   - Choose Save to CSV at upper right to save the report as a CSV file.
     The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the SQL code.

Saving your converted application code with AWS SCT

Use the following procedure to save your converted application code.

To save your converted application code
1. Expand the C++ node under Applications in the target database panel.
2. Choose your converted application, and choose **Save**.
3. Enter the path to the folder to save the converted application code, and choose **Select folder**.

### Converting SQL code in Java applications with AWS SCT

For an Oracle to PostgreSQL conversion, you can use AWS Schema Conversion Tool to convert SQL code embedded into your Java applications. This specific Java application converter understands the application logic. It collects statements that are located in different application objects, such as functions, parameters, local variables, and so on.

Because of this deep analysis, the Java application SQL code converter provides better conversion results compared the generic converter.

### Creating Java application conversion projects in AWS SCT

You can create a Java application conversion project only for converting Oracle database schemas to PostgreSQL database schemas. Make sure that you add a mapping rule in your project that includes a source Oracle schema and a target PostgreSQL database. For more information, see [Creating mapping rules in AWS SCT](#) (p. 122).

You can add multiple application conversion projects in a single AWS SCT project. Use the following procedure to create a Java application conversion project.

#### To create a Java application conversion project

1. Create a database conversion project, and add a source Oracle database. For more information, see [Creating an AWS SCT project](#) (p. 16) and [Adding database servers to an AWS SCT project](#) (p. 17).
2. Add a mapping rule that includes your source Oracle database and a target PostgreSQL database. You can add a target PostgreSQL database or use a virtual PostgreSQL target database platform in a mapping rule. For more information, see [Creating mapping rules in AWS SCT](#) (p. 122) and [Using virtual targets](#) (p. 123).
3. On the **View** menu, choose **Main view**.
4. On the **Applications** menu, choose **New Java application**.

The **Creating a Java application conversion project** dialog box appears.
5. For **Name**, enter a name for your Java application conversion project. Because each database schema conversion project can have one or more child application conversion projects, choose a name that makes sense if you add multiple projects.

6. For **Location**, enter the location of the source code for your application.

7. In the source tree, choose the schema that your application uses. Make sure that this schema is part of a mapping rule. AWS SCT highlights the schemas that are part of a mapping rule in bold.

8. Choose **OK** to create your Java application conversion project.

9. Find your Java application conversion project in the **Applications** node in the left panel.

**Managing Java application conversion projects in AWS SCT**

You can add multiple Java application conversion projects, update the application code in the AWS SCT project, or remove a Java conversion project from your AWS SCT project.

**To add an additional Java application conversion project**

1. Expand the **Applications** node in the left panel.
2. Choose the **Java** node, and open the context (right-click) menu.
3. Choose **New application**.
4. Enter the information that is required to create a new Java application conversion project. For more information, see Creating Java application conversion projects (p. 285).

After you make changes in your source application code, upload it into the AWS SCT project.

**To upload the updated application code**

1. Expand the **Java** node under **Applications** in the left panel.
2. Choose the application to update, and open the context (right-click) menu.
3. Choose **Refresh** and then choose **Yes**.

AWS SCT uploads your application code from the source files and removes conversion results. To keep code changes that you made in AWS SCT and the conversion results, create a new Java conversion project.

**To remove a Java application conversion project**

1. Expand the **Java** node under **Applications** in the left panel.
2. Choose the application to remove, and open the context (right-click) menu.
3. Choose **Delete** and then choose **OK**.

**Converting your Java application SQL code in AWS SCT**

After you add your Java application to the AWS SCT project, convert SQL code from this application to a format compatible with your target database platform. Use the following procedure to analyze and convert the SQL code embedded in your Java application in the AWS Schema Conversion Tool.

**To convert your SQL code**

1. Expand the **Java** node under **Applications** in the left panel.
2. Choose the application to convert, and open the context (right-click) menu.
3. Choose **Convert**. AWS SCT analyzes your source code files, determines the application logic, and loads code metadata into the project. This code metadata includes Java classes, objects, methods, global variables, interfaces, and so on.

In the target database panel, AWS SCT creates the similar folders structure to your source application project. Here you can review the converted application code.
Creating a Java application conversion assessment report in AWS SCT

The Java application conversion assessment report provides information about converting the SQL code embedded in your Java application to a format compatible with your target database. The assessment report provides conversion details for all SQL execution points and all source code files. The assessment report also includes action items for SQL code that AWS SCT can't convert.

Use the following procedure to create a Java application conversion assessment report.

To create a Java application conversion assessment report

1. Expand the Java node under Applications in the left panel.
2. Choose the application to convert, and open the context (right-click) menu.
3. Choose Convert.
4. On the View menu, choose Assessment report view.

4. Save your converted application code. For more information, see Saving your converted application code (p. 289).
5. Review the **Summary** tab.

The **Summary** tab, shown following, displays the executive summary information from the Java application assessment report. It shows conversion results for all SQL execution points and all source code files.

6. Choose **Save statements to JSON** to save the extracted SQL code from your Java application as a JSON file.

7. (Optional) Save a local copy of the report as either a PDF file or a comma-separated values (CSV) file:
   - Choose **Save to PDF** at upper right to save the report as a PDF file.
     The PDF file contains the executive summary, action items, and recommendations for application conversion.
   - Choose **Save to CSV** at upper right to save the report as a CSV file.
     The CSV file contains action items, recommended actions, and an estimated complexity of manual effort required to convert the SQL code.

### Saving your converted application code with AWS SCT

Use the following procedure to save your converted application code.

**To save your converted application code**

1. Expand the **Java** node under **Applications** in the target database panel.
2. Choose your converted application, and choose **Save**.
3. Enter the path to the folder to save the converted application code, and choose **Select folder**.
Converting SQL code in Pro*C applications with AWS SCT

For an Oracle to PostgreSQL conversion, you can use the AWS Schema Conversion Tool (AWS SCT) to convert SQL code embedded into your Pro*C applications. This specific Pro*C application converter understands the application logic. It collects statements that are located in different application objects, such as functions, parameters, local variables, and so on.

Because of this deep analysis, the Pro*C application SQL code converter provides better conversion results compared the generic converter.

Creating Pro*C application conversion projects in AWS SCT

You can create a Pro*C application conversion project only for converting Oracle database schemas to PostgreSQL database schemas. Make sure that you add a mapping rule in your project that includes a source Oracle schema and a target PostgreSQL database. For more information, see Creating mapping rules in AWS SCT (p. 122).

You can add multiple application conversion projects in a single AWS SCT project. Use the following procedure to create a Pro*C application conversion project.

To create a Pro*C application conversion project

1. Create a database conversion project, and add a source Oracle database. For more information, see Creating an AWS SCT project (p. 16) and Adding database servers to an AWS SCT project (p. 17).
2. Add a mapping rule that includes your source Oracle database and a target PostgreSQL database. You can add a target PostgreSQL database or use a virtual PostgreSQL target database platform in a mapping rule. For more information, see Creating mapping rules in AWS SCT (p. 122) and Using virtual targets (p. 123).
3. On the View menu, choose Main view.
4. On the Applications menu, choose New Pro*C application.

The Creating a Pro*C application conversion project dialog box appears.
5. For Name, enter a name for your Pro*C application conversion project. Because each database
schema conversion project can have one or more child application conversion projects, choose a
name that makes sense if you add multiple projects.

6. For Location, enter the location of the source code for your application.

7. In the source tree, choose the schema that your application uses. Make sure that this schema is part
of a mapping rule. AWS SCT highlights the schemas that are part of a mapping rule in bold.

8. Choose OK to create your Pro*C application conversion project.

9. Find your Pro*C application conversion project in the Applications node in the left panel.

Managing Pro*C application conversion projects in
AWS SCT

You can add multiple Pro*C application conversion projects, update the application code in the AWS SCT
project, or remove a Pro*C conversion project from your AWS SCT project.

To add an additional Pro*C application conversion project

1. Expand the Applications node in the left panel.

2. Choose the Pro*C node, and open the context (right-click) menu.

3. Choose New application.

4. Enter the information that is required to create a new Pro*C application conversion project. For more
information, see Creating Pro*C application conversion projects (p. 290).

After you make changes in your source application code, upload it into the AWS SCT project.

To upload the updated application code

1. Expand the Pro*C node under Applications in the left panel.

2. Choose the application to update, and open the context (right-click) menu.
3. Choose **Refresh** and then choose **Yes**.

AWS SCT uploads your application code from the source files and removes conversion results.
To keep code changes that you made in AWS SCT and the conversion results, create a new Pro*C conversion project.

**To remove a Pro*C application conversion project**

1. Expand the **Pro*C** node under **Applications** in the left panel.
2. Choose the application to remove, and open the context (right-click) menu.
3. Choose **Delete** and then choose **OK**.

**Converting your Pro*C application SQL code in AWS SCT**

After you add your Pro*C application to the AWS SCT project, convert SQL code from this application to a format compatible with your target database platform. Use the following procedure to analyze and convert the SQL code embedded in your Pro*C application in the AWS Schema Conversion Tool.

**To convert your SQL code**

1. Expand the **Pro*C** node under **Applications** in the left panel.
2. Choose the application to convert and then choose **Settings**.
   a. For **Global header file path**, enter the path to the header files that your application project uses.
   b. Choose **Interpret all unresolved host variables as** to see all unresolved variables in the converted code.
   c. Choose **Use fixed-width string conversion function from the extension pack** to use the extension pack functions in the converted SQL code. AWS SCT includes the extension pack files in your application project.
   d. Choose **Transform anonymous PL/SQL blocks to standalone SQL calls or stored functions** to create stored procedures in your target database for all anonymous PL/SQL blocks. AWS SCT then includes the runs of these stored procedures in the converted application code.
   e. Choose **Use custom cursor flow** to improve the conversion of Oracle database cursors.
3. In the left panel, choose the application to convert, and open the context (right-click) menu.
4. Choose **Convert**. AWS SCT analyzes your source code files, determines the application logic, and loads code metadata into the project. This code metadata includes Pro*C classes, objects, methods, global variables, interfaces, and so on.

In the target database panel, AWS SCT creates the similar folders structure to your source application project. Here you can review the converted application code.
Creating a Pro*C application conversion assessment report in AWS SCT

The Pro*C application conversion assessment report provides information about converting the SQL code embedded in your Pro*C application to a format compatible with your target database. The assessment report provides conversion details for all SQL execution points and all source code files. The assessment report also includes action items for SQL code that AWS SCT can't convert.

Use the following procedure to create a Pro*C application conversion assessment report.

To create a Pro*C application conversion assessment report

1. Expand the Pro*C node under Applications in the left panel.
2. Choose the application to convert, and open the context (right-click) menu.
3. Choose Convert.
4. On the View menu, choose Assessment report view.
5. Review the Summary tab.

The Summary tab, shown following, displays the executive summary information from the Pro*C application assessment report. It shows conversion results for all SQL execution points and all source code files.
Editing and saving your converted application code with AWS SCT

You can edit the converted SQL statements and use AWS SCT to embed this edited code into the converted Pro*C application code. Use the following procedure to edit your converted SQL code.

**To edit your converted SQL code**

1. Expand the **Pro*C** node under **Applications** in the left panel.
2. Choose the application to convert, open the context (right-click) menu, and choose **Convert**.
3. On the **View** menu, choose **Assessment report view**.
4. Choose **Save statements to CSV** to save the extracted SQL code from your Pro*C application as a CSV file.
5. Enter the name of the CSV file to save the extracted SQL code, and choose **Save**.
6. Edit the extracted SQL code.
7. On the **View** menu, choose **Main view**.
8. Expand the **Pro*C** node under **Applications** in the target database panel.
9. Choose your converted application, open the context (right-click) menu, and choose **Import statements from CSV**.
10. Choose **Yes**, then choose the file with your edited SQL code, and choose **Open**.
AWS SCT breaks the converted SQL statements into parts and places them into the appropriate objects of your source application code. Use the following procedure to save your converted application code.

**To save your converted application code**

1. Expand the **Pro*C** node under **Applications** in the target database panel.
2. Choose your converted application, and choose **Save**.
3. Enter the path to the folder to save the converted application code, and choose **Select folder**.
Using AWS SCT extension packs

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

Each AWS SCT extension pack includes the following components:

- **DB schema** – Includes SQL functions, procedures, and tables for emulating certain online transaction processing (OLTP) and online analytical processing (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 about this library, see Using the custom Python library for AWS SCT extension packs (p. 297).
- **AWS Lambda functions** (for certain OLTP databases) – Includes AWS Lambda functions that emulate complex database functionality, such as job scheduling and sending emails.

You can apply AWS SCT extension packs in two ways:

- AWS SCT can automatically apply an extension pack when you apply a target database script by choosing **Apply to database** from the context menu. AWS SCT applies the extension pack before it applies all other schema objects.
- To manually apply an extension pack, choose the target database and then choose **Apply extension pack for** from the context (right-click) menu. In most situations, automatic application is sufficient. However, you might want to apply the pack manually if it’s accidentally deleted.

Each time that you apply an AWS SCT extension pack 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. To turn off these notifications, choose **Settings**, **Global settings**, **Notifications**, and then select **Hide the extension pack replacement alert**.

For a conversion from Microsoft SQL Server to PostgreSQL, you can use the SQL Server to PostgreSQL extension pack in AWS SCT. This extension pack emulates SQL Server Agent and SQL Server Database Mail. For more information, see Emulating SQL Server Agent in PostgreSQL with an extension pack (p. 79) and Emulating SQL Server Database Mail in PostgreSQL with an extension pack (p. 88).

Following, you can find more information about working with AWS SCT extension packs.

**Topics**

- Using the extension pack schema (p. 296)
- Using the custom Python library for AWS SCT extension packs (p. 297)
- Using the AWS Lambda functions from the AWS SCT extension pack (p. 298)

**Using the extension pack schema**

When you convert your database or data warehouse schema, 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 AWS SCT extension packs

In some cases, AWS Schema Conversion Tool can't convert source database features to equivalent Amazon Redshift features. The relevant 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. 298).

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 the custom Python library

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

### Applying the extension pack to install the custom Python library

Use the following procedure to apply the extension pack to install the custom Python library.

**To apply the extension pack**

1. In the AWS Schema Conversion Tool, in the target database tree, open the context (right-click) menu, choose **Apply extension pack for**, and then choose your source database platform.
AWS SCT provides an extension pack that contains Lambda functions for email, job scheduling, and other features for databases hosted on Amazon EC2.

**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 to support Lambda functions

Use the following procedure to apply the extension pack to support Lambda functions.

**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, choose **Apply extension pack for**, and then choose your source database platform.

   ![Apply Extension Pack for](image)

   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 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 Storing AWS service profiles in the AWS SCT (p. 34).

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.


   When you are done, choose Finish.

   **Note**
   To update an extension pack and overwrite the old extension pack components, be sure to use the latest version of AWS SCT.
Best practices for AWS SCT

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

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 JavaOptions section to set the minimum and maximum memory available. The following example sets the minimum to 4 GB and the maximum to 40 GB,

```
[JavaOptions]
-Xmx48960M
-Xms4096M
```

Increasing logging information

In addition to managing the memory settings of AWS SCT, you can increase the logging information produced by AWS SCT when converting new projects. Although increasing logging information might
slow conversion slightly, the changes can help you provide robust information to AWS Support if errors arise.

To change logging settings

1. On the Settings menu, choose Global Settings.
2. In the Global settings window, choose Logging.
3. For Debug mode, choose True.
4. Choose key items to increase the logging information.

   For example, to help with key problem areas during conversion, set Parser, Type Mapping, and User Interface to TRACE.

You can also configure the log location. If information becomes too verbose for the file system where logs are streaming, change to a location with sufficient space to capture logs.

To transmit logs to AWS Support, go to the directory where the logs are stored, and compress up all the files into a manageable single .zip file. Then upload the .zip file with your support case. When the initial analysis is completed and ongoing development resumes, return Debug mode to false to eliminate the verbose logging. Then increase conversion speed.

Tip
To manage the log size and streamline reporting issues, remove the logs or move them to another location after a successful conversion. Doing this ensures that only the relevant errors and information are transmitted to AWS Support and keeps the log file system from filling.
Troubleshooting issues with AWS SCT

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.

- Cannot load objects tree.
- ORA-00942: table or view does not exist

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_prive and grants the user with this ID the minimum permissions required to convert schema from an Oracle source database.

```sql
create user min_prive identified by min_prive;
grant connect to min_prive;
grant select_catalog_role to min_prive;
grant select any dictionary to min_prive;
```

Assessment report warning message

To assess the complexity of converting to another database engine, AWS SCT requires access to objects in your source database. When SCT can't perform an assessment because problems were encountered during scanning, a warning message is issued that indicates overall conversion percentage is reduced. Following are reasons why AWS SCT might encounter problems during scanning:

- The user account connected to the database doesn't have access to all of the needed objects. For more information about SCT required security permissions and privileges for your database, see Sources for AWS SCT (p. 43) for the appropriate source database section in this guide.
- An object cited in the schema no longer exists in the database. To help resolve the issue, you can connect with SYSDBA permissions and check if the object is present in the database.
- SCT is trying to assess an object that is encrypted.
AWS SCT Reference

Following, you can find reference material for use with the AWS Schema Conversion Tool command line interface (CLI):

AWS Schema Conversion Tool CLI Reference
# Release notes for AWS SCT

This section contains release notes for AWS SCT, starting with version 1.0.640.

## Release notes for AWS SCT Build 662

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Added the ability to automatically create AWS SCT projects for each source database when creating the multiserver assessment report. With this option turned on, AWS SCT can add mapping rules to these projects and save conversion statistics for offline use. For more information, see Creating a multiserver assessment report for database migration (p. 136).</td>
</tr>
<tr>
<td>All</td>
<td>Aurora MySQL</td>
<td>Implemented support of the percent (%) as a wildcard in database and schema names when creating the multiserver assessment report.</td>
</tr>
<tr>
<td>All</td>
<td>Aurora PostgreSQL</td>
<td>Updated the runtime of all AWS Lambda functions to Python version 3.9.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Upgraded all data extraction agents to use AWS SDK for Java 2.x.</td>
</tr>
<tr>
<td>Azure SQL Database</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of DELETE statements with NOT EXISTS clauses.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Azure Synapse</td>
<td>Amazon Redshift</td>
<td>Resolved an error where the connection to a source database failed.</td>
</tr>
<tr>
<td>Analytics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM Db2 for z/OS</td>
<td>Aurora PostgreSQL</td>
<td>Resolved an error where the converted code of a trigger included two mentions of the object alias.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of objects with names in mixed-case when the Treat database object name as case sensitive option is turned on.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Amazon Redshift</td>
<td>Implemented conversion of the PIVOT and UNPIVOT relational operators.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Amazon Redshift</td>
<td>Implemented conversion of the TIME data type.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora MySQL</td>
<td>Implemented the <code>UTL_TCP.CRLF</code> package constant conversion.</td>
</tr>
<tr>
<td></td>
<td>Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MySQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Fixed an extension pack issue where the length of data types for columns of variable length wasn't maintained during conversion.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Implemented SQL code conversion in C++ applications. For more information, see [Converting SQL code in C++ applications with AWS SCT](p. 279).</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Implemented support of case sensitive naming for the conversion of global variables and associative arrays.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of the <code>TO_CHAR</code>, <code>TO_DATE</code>, and <code>TO_NUMBER</code> functions in the extension pack.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of the <code>TABLE()</code> operator.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Added support for conversion of primary keys and other constraints.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Fixed a problem where action item 12054 doesn't appear during the conversion of conditional statements.</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>Aurora PostgreSQL</td>
<td>Resolved an error when an object with an empty name was created in the target tree during the conversion of tables with columns of user-defined type.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>SAP ASE</td>
<td>Aurora PostgreSQL</td>
<td>Fixed a loader error for stored objects such as scripts, routines, and so on.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
<td>Fixed a problem where action item 22152 doesn't appear when required and AWS SCT displays the conversion result as a comment.</td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the date and time functions, implemented support of time zones.</td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where non-recursive common table expressions (CTEs) with a <code>WITH</code> clause were converted as recursive CTEs.</td>
</tr>
</tbody>
</table>
### Teradata
- **Amazon Redshift**
  - Improved conversion of `UPDATE` statements with table links in condition.
- **Amazon Redshift**
  - Improved conversion of `RENAME TABLE` statements.
- **Amazon Redshift**
  - Resolved an issue where empty columns appeared in the comma-separated value (CSV) file with an assessment report.
- **Teradata**
  - Fixed an error where a semicolon was missing in the end of the converted a Basic Teradata Query (BTEQ) macro.
- **Teradata**
  - Improved conversion of multiple data type values in `CASE` statements.
- **Teradata**
  - Improved conversion of the `LIKE ANY` clause with an `ESCAPE` character.
- **Teradata**
  - Improved conversion of the `CAST` function in `INSERT` statements.
- **Teradata**
  - Improved conversion of the time zones, implemented time zone region mapping.
- **Teradata**
  - Resolved an issue where action item 9998 unexpectedly appears during the conversion of shell scripts with BTEQ scripts.
- **Teradata**
  - Implemented the limit of 500 characters for the values of substitution variables.
- **Vertica**
  - **Amazon Redshift**
    - Implemented conversion of the `BINARY`, `VARBINARY`, `LONG BINARY`, `BYTEA`, and `RAW` data types to the `VARBYTE` data type.
- **Vertica**
  - **Amazon Redshift**
    - Improved conversion of the built-in functions and literals.

### Vertica
- **Amazon Redshift**
  - Implemented conversion of the `BINARY`, `VARBINARY`, `LONG BINARY`, `BYTEA`, and `RAW` data types to the `VARBYTE` data type.

### All
- **All**
  - Added filters to search for mapping rules in the mapping view. When you apply a filter, AWS SCT displays rules that match the filtering conditions in the Server mappings list. For more information, see Managing mapping rules (p. 122).
- **All**
  - Upgraded Apache Log4j to version 2.17.1.
### What's new, enhanced, or fixed

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Added support of data migration to Amazon Redshift using the ENCRYPTED clause in the COPY command.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Enhanced the REST API of the data extraction agents. The updated REST API adds support of new properties such as encryption key, encryption type, and so on.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Implemented role assuming in the data extraction agents. This update improves the distribution of subtasks, and enables AWS SCT to assign tasks to free agents of the specified role.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Implemented a check that all required components are installed before the extension pack is applied to Amazon Redshift.</td>
</tr>
<tr>
<td>Azure Synapse Analytics</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the ERROR_LINE, ERROR_MESSAGE, ERROR_NUMBER, ERROR_PROCEDURE, ERROR_SEVERITY, and ERROR_STATE system functions for error handling.</td>
</tr>
<tr>
<td>Azure Synapse Analytics</td>
<td>Amazon Redshift</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server DW</td>
<td>Amazon Redshift</td>
<td></td>
</tr>
<tr>
<td>IBM Db2 for z/OS</td>
<td>Aurora MySQL</td>
<td>Added support of IBM Db2 for z/OS version 12 as a source for database migration projects in AWS SCT. For more information, see Using IBM Db2 for z/OS as a source (p. 48).</td>
</tr>
<tr>
<td>IBM Db2 for z/OS</td>
<td>Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>IBM Db2 for z/OS</td>
<td>MySQL</td>
<td></td>
</tr>
<tr>
<td>IBM Db2 for z/OS</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>IBM Db2 LUW</td>
<td>All</td>
<td>Enhanced the source metadata loader to ensure that AWS SCT loads routine parameters that duplicate column names.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>Aurora PostgreSQL</td>
<td>Fixed a transformer error for procedures with the SET NOCOUNT ON set statement.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>MySQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of the CONCAT function when an input value is a variable of the user-defined type.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Resolved an issue where the DATEPART function was incorrectly converted.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Babelfish for Aurora PostgreSQL</td>
<td>Implemented support of the new version of the Babelfish features configuration file. This file defines SQL features that are supported and not supported by specific Babelfish versions.</td>
</tr>
<tr>
<td>Microsoft SQL Server DW</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where procedures with an EXECUTE statement were incorrectly converted.</td>
</tr>
<tr>
<td>Microsoft SSIS</td>
<td>AWS Glue</td>
<td>Improved the user interface of the job configuration wizard. AWS SCT now displays only available connections in the connection configuration section.</td>
</tr>
<tr>
<td>Microsoft SSIS</td>
<td>AWS Glue</td>
<td>Resolved an issue where the transformation rules weren't applied to package tasks and variable rules.</td>
</tr>
<tr>
<td>Microsoft SSIS</td>
<td>AWS Glue Studio</td>
<td>Added a new action item 25042 for unsupported components.</td>
</tr>
<tr>
<td>Microsoft SSIS</td>
<td>AWS Glue Studio</td>
<td>Implemented conversion of Microsoft SQL Server Integration Services (SSIS) extract, transform, and load (ETL) packages to AWS Glue Studio. For more information, see Converting SSIS to AWS Glue Studio (p. 191).</td>
</tr>
<tr>
<td>Oracle</td>
<td>MariaDB</td>
<td>Fixed a problem with conversion of the MINUS operator.</td>
</tr>
<tr>
<td>Oracle</td>
<td>MariaDB</td>
<td>Improved conversion of the ROWNUM, SYS_GUID, TO_CHAR, and ADD_MONTHS functions when the sql_mode system variable in MariaDB is to Oracle.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Added an option to avoid conversion of bind variables types to SQL types in generic application conversion projects.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Added an option to avoid adding the schema name to the name of the converted object in generic application conversion projects.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Added support of the ?x bind variable format for application SQL code conversion.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Implemented conversion of the RAW data type to the VARBYTE data type.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added an option to emulate SET tables in the converted code. For this emulation, AWS SCT supports MIN and MAX conditions.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of join operations that have parameters of different data types. This update enables AWS SCT to apply transformation rules during conversion of such operations.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where the GROUP BY clause was incorrectly converted.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where the QUALIFY clause was incorrectly converted.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an unexpected error occurred during FastExport scripts import.</td>
</tr>
</tbody>
</table>
### What's new, enhanced, or fixed

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Implemented the ability to edit the values of variables in Teradata BTEQ and shell scripts.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where the manifest script was missing for the converted Teradata FastLoad sessions.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where the extension of the manifest file was missing in the uniform resource locator (URL) for the converted FastLoad scripts.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed a loader error for scripts with substitution variables.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed a problem where action item 27022 doesn't appear when required.</td>
</tr>
</tbody>
</table>

### Release notes for AWS SCT Build 660

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Added support of AWS Secrets Manager and Secure Sockets Layer (SSL) in the multiserver assessment report. For more information, see Creating a multiserver assessment report for database migration (p. 136).</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Improved statistics collection for converted objects.</td>
</tr>
<tr>
<td>All</td>
<td>PostgreSQL</td>
<td>Implemented support of PostgreSQL major version 14 and MariaDB 10.6 as migration targets.</td>
</tr>
<tr>
<td>Azure Synapse Analytics</td>
<td>Amazon Redshift</td>
<td>Improved transformation logic for the names of converted objects.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of the XML data type.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Resolved an issue where NOT LIKE clauses were incorrectly converted.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>Aurora PostgreSQL</td>
<td>Fixed a transformer error for procedures with INSERT, DELETE, and UPDATE statements that include the OUTPUT clause.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Fixed a transformer error for procedures with the <code>RETURN @@ROWCOUNT</code> statement.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td>Improved conversion of procedures that use linked servers.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>All</td>
<td>Added support of Microsoft Windows Authentication in the multiserver assessment report.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Fixed a transformer error for table value constructors.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td>Fixed a transformer error for table value constructors.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Amazon Redshift and AWS Glue</td>
<td>Improved conversion of extract, transform, and load (ETL) scripts to include the correct path to the converted scripts.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where different converted scripts were generated for virtual and real target database platforms.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Added support for conversion of indexes for materialized views.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Fixed a problem where action item 5982 doesn't appear when converting <code>PRIMARY KEY</code> and <code>UNIQUE</code> constraints with the <code>NOVALIDATE</code> option.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where additional categories were displayed in the converted schema.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a problem where action item 13185 doesn't appear when converting an unresolved column as an argument of the <code>CAST</code> function.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of <code>DELETE</code> and <code>DELETE ALL</code> statements to use the <code>TRUNCATE</code> command in the converted code.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of <code>SET</code> tables.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of <code>NORMALIZE</code> condition.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Updated the assessment report to remove the database schema conversion statistics from the list of database storage objects.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>UPDATE</code> statement without the <code>FROM</code> clause.</td>
</tr>
</tbody>
</table>
### What's new, enhanced, or fixed

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Implemented support of the <code>VARBYTE</code> data type in the converted code.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>AWS Glue</td>
<td>Resolved an issue where the <strong>Convert to AWS Glue</strong> option was deactivated in the context menu.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where data types were missing in the converted code.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where substitution variables were incorrectly quoted in the converted code.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed a problem with conversion of substitution variables with values in FastLoad scripts.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Implemented support of the <code>TIME</code> data type in the converted code.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Improved conversion of <code>SELECT DISTINCT</code> and <code>ORDER BY</code> expressions.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Added support for conversion of constraints.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Resolved an error where an assessment report wasn't saved as a comma-separated value (CSV) file.</td>
</tr>
</tbody>
</table>

### Release notes for AWS SCT Build 659

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Improved the <strong>New project wizard</strong> that generates a combined assessment report for multiple source databases.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Fixed an issue where the extension pack wasn't created in projects that include multiple source and target databases.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Improved conversion of SQL code that is embedded in application source code.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added the ability to run scripts from different folders in the AWS SCT command-line interface.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Improved the warning message provided when users choose <strong>Run optimization</strong> in migration projects with the Amazon Redshift virtual target database platform.</td>
</tr>
<tr>
<td>All</td>
<td>Aurora PostgreSQL</td>
<td>Implemented support of PostgreSQL major version 13 on Aurora PostgreSQL-Compatible Edition as a migration target.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon RDS for MySQL</td>
<td>Implemented the case insensitive code conversion by default.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Azure Synapse Analytics</td>
<td>Amazon Redshift</td>
<td>Resolved an error where the connection to a source database failed in the command-line interface.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td>Improved conversion of procedures that include UPDATE statements with join conditions.</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td>Improved conversion of triggers, stored procedures, and functions that include the value after the equal sign.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>Fixed a transformer error for procedures with the DELETE statement and the OR operator.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td>Improved conversion of the OUTPUT clause.</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td>Improved conversion of the NUMERIC data type.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>Improved conversion of views which have a table alias with same name as the original table.</td>
</tr>
<tr>
<td>Microsoft SQL Server DW</td>
<td>Amazon Redshift and</td>
<td>Added the ability to repeat the run of change data capture (CDC) data migration tasks every day.</td>
</tr>
<tr>
<td></td>
<td>AWS Glue</td>
<td>Fixed an issue where the Tasks tab becomes inactive after un-registering a data extraction agent.</td>
</tr>
<tr>
<td>Microsoft SSIS</td>
<td>AWS Glue</td>
<td>Fixed an issue where the confirmation of the registration of the data migration agent didn't display in the user interface.</td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where a connection to a source database failed with a Loader error.</td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
<td>Resolved an error where data migration agents failed to run after opening a saved project.</td>
</tr>
<tr>
<td>Oracle RDS for Oracle</td>
<td>Amazon RDS for Oracle</td>
<td>Implemented support of Oracle Unified Auditing.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Implemented SQL code conversion in C# applications. For more information, see Converting SQL code in C# applications (p. 275).</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Implemented a new transformation logic for case-sensitive object names to improve the visibility of code conversion changes. AWS SCT converts object names in uppercase to lowercase. The opposite is also true; AWS SCT converts object names in lowercase to uppercase. Other object names and reserved words are converted without changes.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgresQL</td>
<td>Improved conversion of hash partitions without the <code>NOT NULL</code> constraint.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgresQL</td>
<td>Added support for conversion of Oracle <code>CHECK</code>, <code>FOREIGN KEY</code>, and <code>NOT NULL</code> constraints with the <code>ENABLE NOVALIDATE</code> clause.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where the incorrect values for floating point numbers were migrated.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift and AWS Glue</td>
<td>Resolved an issue with empty columns in the database migration assessment report in a comma-separated value (CSV) file.</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>PostgresQL</td>
<td>Fixed an issue with an unexpected conversion interruption.</td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>VARIANT</code> data type.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>COLLECT STATISTICS</code> statement.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a problem where action item 9998 doesn't appear when converting nested views with <code>PERIOD</code> columns.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift and AWS Glue</td>
<td>Fixed an issue where a virtual AWS Glue target platform didn't display in the UI after opening a saved project.</td>
</tr>
<tr>
<td>Teradata</td>
<td>AWS Glue</td>
<td>Fixed an issue where the conversion to a virtual AWS Glue target platform wasn't supported after opening a saved project.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved syntax highlighting of the converted code.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Implemented checking parameter values after upload. Unsupported values are highlighted on the <code>Variables</code> tab.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Implemented conversion of aggregate functions.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Implemented conversion of projections to materialized views and improved the UI that displays the source code of projections.</td>
</tr>
</tbody>
</table>
## Release notes for AWS SCT Build 658

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Provided integration with AWS Secrets Manager. You can now use database connection credentials that are stored in Secrets Manager.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added support for scripts in the YAML format in the AWS SCT command-line interface.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Implemented support of Amazon S3 interface endpoints (VPCE) in data extraction agents.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Added support for the Amazon Redshift virtual target database platform in addition to the already supported Amazon Redshift and AWS Glue combination.</td>
</tr>
<tr>
<td>Greenplum</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where the Save as SQL option didn't save converted SQL code to a file.</td>
</tr>
<tr>
<td>IBM Db2 LUW</td>
<td>Aurora MySQL</td>
<td>Improved conversion to support new features of Amazon Aurora MySQL-Compatible Edition with MySQL 8.0 compatibility.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>Aurora MySQL</td>
<td>Fixed a problem where action item 810 doesn't appear when required.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora MySQL</td>
<td>Improved conversion of procedures with UPDATE, DELETE, and INSERT statements.</td>
</tr>
<tr>
<td>Oracle</td>
<td>MySQL</td>
<td>Fixed a problem where action item 7810 doesn't appear when required.</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>PostgreSQL</td>
<td>Improved conversion of an EXEC statement that is nested inside an IF . . . ELSE statement.</td>
</tr>
</tbody>
</table>

**Version 1.0**

315
<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora, PostgreSQL, PostgreSQL</td>
<td>Improved conversion of indexed views.</td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
<td>Improved data migration agents by tracking live transactions during full load in the change data capture (CDC) operation. You can now stop data migration tasks if the CDC session is scheduled to start at a certain time. Also, you can see the error logging level in the console after you stop a task with CDC.</td>
</tr>
<tr>
<td>Oracle</td>
<td>All</td>
<td>Enhanced the table loader to ensure that AWS SCT loads objects with sharing options.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora, PostgreSQL, PostgreSQL</td>
<td>Improved conversion of the <code>SYSDATE</code> function and added the ability to change the time zone in Conversion settings.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora, PostgreSQL, PostgreSQL</td>
<td>Resolved an issue where dynamic statements weren't converted.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora, PostgreSQL, PostgreSQL</td>
<td>Fixed an issue where the converted code doesn't include system-generated names.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora, PostgreSQL, PostgreSQL</td>
<td>Improved conversion of <code>SELECT</code> statements that are nested inside triggers.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Aurora, PostgreSQL, PostgreSQL</td>
<td>Improved conversion of the <code>TO_DATE</code>, <code>TO_TIMESTAMP</code>, and <code>TO_TIMESTAMP_TZ</code> functions in the extension pack.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Added an option to save converted SQL code in different files for each object or for each statement.</td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
<td>Added an option to save converted SQL code in different files for each object or for each statement.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>CONCAT</code> function.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of a <code>SELECT</code> statement that is nested inside a <code>WHERE</code> clause.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue when <code>SET</code> and <code>MULTISET</code> tables were incorrectly converted after users drop and recreate a table.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the procedures that include a <code>WITH</code> clause.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>DATE</code> data type.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where an unexpected transformer error occurred during FastExport scripts conversion.</td>
</tr>
</tbody>
</table>
### What's new, enhanced, or fixed

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Added support for conversion of a join index to a materialized view.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Added support for conversion of a <code>TITLE</code> definition that includes multiple lines.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where the size of a geospatial data type wasn’t converted.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed a problem where the parameter names were converted to lowercase characters.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed an issue when a stored procedure that is nested inside a <code>MACRO</code> statement wasn’t converted.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>ALL</code> operator.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where the <code>Use Union all view?</code> option in <code>Conversion settings</code> wasn’t applied.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>TIME</code> and <code>TIME WITH TIMEZONE</code> data types.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Resolved an issue with loading of flex tables.</td>
</tr>
</tbody>
</table>

### Issues resolved:
- General improvements.

---

**Release notes for AWS SCT Build 657**

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Upgraded Apache Log4j to version 2.17 to address security vulnerability issues.</td>
</tr>
<tr>
<td>All</td>
<td>Amazon Redshift</td>
<td>Improved schema optimization projects, where key management statistics weren't saved in the AWS SCT project.</td>
</tr>
<tr>
<td>Amazon Redshift</td>
<td>Amazon Redshift</td>
<td>Fixed a problem with the server information update.</td>
</tr>
<tr>
<td>Apache Cassandra</td>
<td>Amazon DynamoDB</td>
<td>Fixed an issue with mapping rules when using the AWS SCT command-line interface.</td>
</tr>
<tr>
<td>Apache Cassandra</td>
<td>Amazon DynamoDB</td>
<td>Resolved an issue when the migration task wasn't created because of an updated title in the certificate.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Fixed a problem so that action item 7672 doesn't appear during the conversion of Microsoft SQL Server procedures with dynamic SQL.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Azure SQL Database</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of table-valued functions.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Resolved an issue where the OUT argument in a stored procedure with the default return value wasn't converted to the INOUT argument.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Greenplum Database</td>
<td>Amazon Redshift</td>
<td>Improved optimization strategies by finding the most used tables and columns from the QueryLog table.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Fixed problems with conversion of the following:</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>• String concatenation assignment operator (+=)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SCOPE_IDENTITY function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• varchar(max) data type</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of views with unsupported functions.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed an issue where unsupported functions as an argument to another function were incorrectly converted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Babelfish for Aurora PostgreSQL</td>
<td>Improved conversion of transition table references.</td>
</tr>
<tr>
<td></td>
<td>Amazon Redshift</td>
<td>Added the aggregate functions category to the source database metadata tree.</td>
</tr>
<tr>
<td></td>
<td>Redshift</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improved conversion of the TIME data type.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Azure Synapse Analytics</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where DROP and CREATE scripts weren't saved when using a virtual target database platform.</td>
</tr>
<tr>
<td>Greenplum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server DW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Snowflake</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teradata</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server Integration Services</td>
<td>AWS Glue</td>
<td>Resolved an issue where the scripts of source objects didn't display in the UI.</td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
<td>Improved optimization strategies by choosing the fact table and appropriate dimensions for collocation.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Resolved an issue to correctly convert Oracle triggers, which use sequence numbers.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of views with public database links.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Improved optimization strategies by analyzing the cardinality of index columns.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where custom user-defined scalar functions with string concatenation were incorrectly converted.</td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where the <strong>Save as SQL</strong> option didn't display in the UI.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where statistic collection failed with the <strong>LOADER_ERROR</strong> exception.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where the <strong>Create report</strong> option didn't display in the UI.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <strong>CAST</strong> function.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a broken conversion for <strong>ST_Line_Interpolate_Point</strong>.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Removed an unexpected value from the Python library path.</td>
</tr>
</tbody>
</table>
## Source | Target | What's new, enhanced, or fixed
--- | --- | ---
Teradata | Amazon Redshift | Fixed a resolver error that appeared during the conversion of multiple FastLoad scripts.
Teradata BTEQ | Amazon Redshift RSQL | Improved conversion of the `DATABASE` command and geometry data types.
Teradata BTEQ | AWS Glue | Fixed an issue with an incorrect synchronization of the source and target scripts in the UI.

### Issues resolved:
- General improvements.

---

## Release notes for AWS SCT Build 656

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Added support of multiple source and target databases within one project. Users can now create mapping rules to match different database schemas and target platforms in the same project.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added support of virtual target database platforms. Users now don't need to connect to a target database to see how AWS SCT converts their source database schema.</td>
</tr>
</tbody>
</table>
| All | All | **UI improvements:**  
  - Added the **Connect to the server** and **Disconnect from the server** options to the source and target metadata trees.  
  - Added an option to remove a database server from the AWS SCT project. |
<p>| Cassandra | Amazon DynamoDB | Resolved a search issue where the <code>CASSANDRA_HOME</code> variable didn't include a slash (<code>/</code>) after <code>cassandra.yaml</code> or the <code>conf</code> folder. |
| Cassandra | Amazon DynamoDB | Added support of the Amazon Machine Image (AMI) for Amazon Linux 2. |
| Cassandra | Amazon DynamoDB | Improved error message provided when an incorrect key is given for Cassandra. |
| Cassandra | Amazon DynamoDB | Improved conversion by changing a property in the <code>cassandra-env.yaml</code> file depending on the version of the target database. |
| Cassandra | Amazon DynamoDB | Increased the Java version on the target Cassandra Datacenter to 1.8.0. |
| Greenplum | Amazon Redshift | Improved optimization strategies in <strong>Project Settings</strong>. |</p>
<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greenplum</td>
<td>Amazon Redshift</td>
<td>Resolved a data migration issue where objects weren't applied to database with this error: An I/O error occurred while sending to the backend.</td>
</tr>
<tr>
<td>Greenplum, Microsoft SQL Server DW</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where the Apply RTRIM to string columns option didn't display in the UI.</td>
</tr>
<tr>
<td>Microsoft SQL Server DW</td>
<td>Babelfish for Aurora PostgreSQL</td>
<td>Added support for Babelfish for Aurora PostgreSQL as a target platform. Users can now create an assessment report to estimate the migration from SQL Server to Babelfish for Aurora PostgreSQL.</td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
<td>Improved optimization strategies in Project Settings.</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>Aurora PostgreSQL</td>
<td>Implemented the ability to generate unique names for indexes.</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>Aurora PostgreSQL</td>
<td>Fixed an issue with a duplicate index column in the target script.</td>
</tr>
<tr>
<td>Snowflake</td>
<td>Amazon Redshift</td>
<td>Resolved a problem where Hide empty schemas, Hide empty databases, and Hide system databases/schemas options weren't displayed in the UI.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added support for conversion of Teradata MultiLoad job scripts to Amazon Redshift RSQL scripts.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed a problem with conversion of substitution variables in FastLoad and FastExport scripts.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed an issue where action items didn't display in the Action Items tab after switching from the Summary tab.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where an error occurs after generating report during FastExport scripts conversion.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved formatting issues after shell scripts conversion.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Fixed a problem so that AI 13177 is now commented in converted script.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a broken conversion of temporal tables.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of the <code>SET QUERY_BAND</code> statement.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a broken conversion of the <code>NORMALIZE</code> operation.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Improved the description of AI 17008.</td>
</tr>
</tbody>
</table>

**Issues resolved:**

- General improvements.

---

**Release notes for AWS SCT Build 655**

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a problem to ensure all assessment issues appear in reports when FastLoad or MultiLoad is used.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added support for conversion of Teradata FastExport job scripts to Amazon Redshift RSQL scripts.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a problem to ensure the Save manifest to S3 action is enabled in offline mode when using FastLoad.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed an issue to ensure mapping rules are applied for scripts like FastLoad.</td>
</tr>
<tr>
<td>Greenplum</td>
<td>Amazon Redshift</td>
<td>Increased the minimum supported driver version for Greenplum to 42.2.5.</td>
</tr>
<tr>
<td>Greenplum</td>
<td>Amazon Redshift</td>
<td>Added a connection to Greenplum via SSL with driver version 42.2.5 or higher.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Improved support for executing custom user-defined scalar functions (UDF) within another UDF.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Fixed an issue where functions weren't applied to database with this error: <code>Failed to compile udf</code>.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Improved conversion by using appropriate type declarations such as, <code>pls-type</code> for <code>%ROWTYPE</code> parameters.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where information type assessment issues didn't display in the report.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved a transformer error after converting some scripts.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a problem so that an issue is now commented in converted script.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where FastExport -&gt;EXPORT -&gt; 'null' displayed instead 'CAST' after conversion.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved a problem where some functions of an extension pack failed when applied with Cause:[JDBC Driver]String index out of range: 0 if using driver version 1.2.43</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>SET table conversion—SET table emulation added for insert-select statements.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>CAST—support additional data type casting.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Fixed a broken conversion for &quot;other_current_time_01&quot;</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Teradata FastExport – Amazon Redshift RSQL: Improved conversion of Teradata FastExport commands—FIELD</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Teradata FastExport – Amazon Redshift RSQL: Improved conversion of Teradata FastExport commands—LAYOUT</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Resolved an issue where target script of objects with SAVE EXCEPTIONS STATEMENT changed after reconversion.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Resolved an issue where wrong field was specified in the ORDER BY clause after proc_cursor_with_calc_columns conversion.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Resolved: an extra aws_oracle_ext$array_id$temporary variable declaration is required in an ASSOCIATIVE COLLECTION conversion.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora PostgreSQL</td>
<td>Resolved: the wrong conversion of a PRIMARY KEY with the same name of an INDEX owned by the same table.</td>
</tr>
</tbody>
</table>

**Issues resolved:**

- General improvements.
### Release notes for AWS SCT Build 654

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Resolved an issue with Hierarchical Query Pseudocolumns, PRIOR columns parsing error.</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>Resolved an issue to correctly convert a multi-line comment containing a slash and asterisk (/*).</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>Added system view USER_COL_COMMENTS emulation to the extension pack.</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>Improved conversion of quoted literals.</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>Improved conversion of LABEL statements that add or replace labels in the descriptions of tables, views, aliases, or columns.</td>
</tr>
<tr>
<td>DB2 LUW</td>
<td>PostgreSQL</td>
<td>Improved conversion of LABEL statements that add or replace labels in the descriptions of tables, views, aliases, or columns.</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>None</td>
<td>Substituted SYS.USER$ system table with DBA_USERS view, and improved queries.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Updated Oracle DW metadata queries.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added support for conversion of shell, Teradata FastLoad, and Teradata Basic Teradata Query (BTEQ) scripts to Amazon Redshift RSQL scripts.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>RSQL</td>
</tr>
<tr>
<td>BTEQ</td>
<td>Amazon Redshift</td>
<td>Resolved issue where &quot;merge_01&quot; was incorrectly converted.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved issue so that End or Identify (EOI) appears at the end of a script on a new line.</td>
</tr>
<tr>
<td>BTEQ</td>
<td>Amazon Redshift</td>
<td>RSQL</td>
</tr>
<tr>
<td>Azure Synapse</td>
<td>Amazon Redshift</td>
<td>Improved error message provided when an incorrect password given for Azure Synapse.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved UPDATE statement conversion to carry forward the right alias name per Teradata standard.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved a cursor conversion error where actions weren't received.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where a TD_NORMALIZE_OVERLAP conversion was dropping rows.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Now supports strict date checking for the enhanced TO_DATE function.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of Built-in function TO_NUMBER(n).</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where the Schemas category was absent from metadata tree.</td>
</tr>
<tr>
<td>Greenplum</td>
<td>Amazon Redshift</td>
<td>Added GP_SEGMENT_ID selection to list when creating virtual partition for a Greenplum table.</td>
</tr>
<tr>
<td>Greenplum</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where functions weren't applied on target.</td>
</tr>
<tr>
<td>MS SQL Server DW</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where a transform error occurs after conversion without AI 9996.</td>
</tr>
<tr>
<td>MS SQL Server DW</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where an error was logged when opening the extension pack wizard.</td>
</tr>
<tr>
<td>MS SQL Server DW</td>
<td>Amazon Redshift</td>
<td>Resolved an issue when an incorrect style of comments was used for Redshift Python functions.</td>
</tr>
<tr>
<td>Netezza</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where a Netezza–Redshift extension pack with an AWS profile failed to create.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Improved conversion of FastLoad SESSIONS command.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Improved FastLoad scripts assessment reports.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Implemented FastLoad WRITER Save to S3 action.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where FastLoad Save Script\Save manifest to s3 buttons weren't active.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where FastLoad multifile_script only created one manifest file after conversion instead of the expected three files.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where FastLoad had extra folders displayed in an S3 path.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift RSQL</td>
<td>Resolved an issue where FastLoad had the incorrect name of the manifest file in an S3 path.</td>
</tr>
</tbody>
</table>
## Issues resolved:

- General improvements.

### Release notes for AWS SCT Build 653

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Implemented the ability to convert dynamic SQL created in called functions or procedures.</td>
</tr>
<tr>
<td></td>
<td>Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Improved Dynamic SQL conversion: In-parameters as bind variables.</td>
</tr>
<tr>
<td></td>
<td>Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle DW 18, 19</td>
<td>Amazon Redshift</td>
<td>Oracle to Redshift conversion improvements implemented: enhanced conversion built-ins. Aggregate LISTAGG; Analytic LISTAGG.</td>
</tr>
<tr>
<td>Oracle DW 18, 19</td>
<td>Amazon Redshift</td>
<td>Oracle to Redshift conversion improvements implemented: Query new features.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>Vertica to Redshift conversion improvements implemented: SSL to JDBC connection with SSL=true.</td>
</tr>
<tr>
<td>MS SQL Server DW</td>
<td>Amazon Redshift</td>
<td>MS SQL Server to Redshift conversion improvements: External Tables.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Teradata to Redshift conversion improvements: INTERVAL data types arithmetic operations.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Teradata to Redshift conversion improvements: Support for lateral column aliases.</td>
</tr>
<tr>
<td>Oracle</td>
<td>None</td>
<td>The following Loader queries now use DBA_USERS instead of SYS.USER $:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• get-tree-path-list-by-name-path.sql</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• estimate-table-or-view-constraints-by-schema.sql</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• estimate-table-or-view-constraints-by-selected-schemas.sql</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved alignment of comments when SCT converts Teradata macros to Redshift stored procedures.</td>
</tr>
<tr>
<td>Oracle DW 18, 19</td>
<td>Amazon Redshift</td>
<td>Improved conversion of Date/Timestamp format elements: TO_DATE, TO_TIMESTAMP, and TO_TIMESTAMP_TZ</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved Teradata cursor conversion error.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved issue that caused attributes of TD_NORMALIZE_OVERLAP to be dropped during conversion.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where MAX function was ignored when SCT converted a query.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>SCT now converts Teradata CHARACTERS function to Redshift LENGTH function.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>SCT now supports conversion of FORMAT to TO_CHAR for most commonly used formats.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Improved conversion of encrypted routines.</td>
</tr>
</tbody>
</table>

**Issues resolved:**

- General improvements.

## Release notes for AWS SCT Build 652

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td>Added app locking for sp_getapplock and sp_releaseapplock functions.</td>
</tr>
<tr>
<td>None</td>
<td>Amazon Redshift</td>
<td>Command Line Interface (CLI) improvement: implemented Script Command mode.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Implemented routine parameters sampling inside dynamic SQL.</td>
</tr>
<tr>
<td>Aurora</td>
<td>PostgreSQL</td>
<td>Conversion improvements to dynamic SQL created in called functions or procedures.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora</td>
<td>Each lambda function is deployed and configured via policy only once, and common lambda functions are reused for all possible sources.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Resolved issue that caused error message, &quot;9996 — Severity critical — Transformer error occurred&quot; when using DB2 LUW as source.</td>
</tr>
<tr>
<td>DB2 LUW</td>
<td>PostgreSQL</td>
<td>Support for recursive table expressions in forthcoming Amazon Redshift launch.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Support Time Zone conversion from Teradata macros to Redshift stored procedures.</td>
</tr>
<tr>
<td>Azure Synapse</td>
<td>Amazon Redshift</td>
<td>Implemented schema optimization rules.</td>
</tr>
</tbody>
</table>

Version 1.0

327
<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Support conversion of Teradata recursive common table expressions (RECURSIVE CTE).</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Support case sensitive identifiers via the user setting, enable_case_sensitive_identifier. So, &quot;COLUMN_NAME&quot; and &quot;Column_Name&quot; become different column names.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved Decimal data type issue so that Decimal fields convert with the same precision.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved issue with interval arithmetic conversion so that interval arithmetic subtraction converts correctly.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved Teradata NUMBER to DATE type casting.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved Teradata DATE to NUMBER type casting</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved PERIOD data type conversion.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved issue with loading metadata for a table with GEOMETRY columns so that it now loads from Teradata correctly.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Support conversion of merge statements when converting Teradata macros to Redshift stored procedures.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of simple macros when migrating from Teradata to Redshift.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Ensured the conversion of Teradata UPDATE statements carry forward the right alias name per Teradata standard.</td>
</tr>
</tbody>
</table>

**Issues resolved:**
- General improvements.

### Release notes for AWS SCT Build 651

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Enhanced AWS SCT reports to update links to the recommended conversion action items listed.</td>
</tr>
<tr>
<td>MS SQL Server</td>
<td>PostgreSQL</td>
<td>Added support for conversion of STR() function.</td>
</tr>
<tr>
<td>MS SQL Server</td>
<td>PostgreSQL</td>
<td>Added support for converting the bitwise EXOR operator (^ in Microsoft SQL Server) to PostgreSQL as the # operator.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Resolved an issue where the AWS SCT extension pack aws_oracle_ext.UNISTR(null) function hung for NULL on a PostgreSQL target. AWS SCT now handles the NULL.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What’s new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift RSQL</td>
<td>Conversion improvements made to resolve an issue where conversion of Amazon Redshift RSQL MERGE gave a transform error.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Implemented enhanced built-ins.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Added metadata feature driven enhancements, including Auto-List partitioning (TBL_PART_LIST_AUTO), Multi-column List (TBL_PART_MULTI_LIST) and Interval-Reference (TBL_PART_RANGE_INTERVAL_REF).</td>
</tr>
<tr>
<td>none</td>
<td>Amazon Redshift</td>
<td>Increased partition table limits of physical partitions used for UNION ALL conversions.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Conversion improvements made to the scope of Assessment reports.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Conversion improvements made to complex Teradata MACRO conversions.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of Teradata macros to Amazon Redshift stored procedures while commenting out unsupported SQL.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where conversion of Teradata macros to Amazon Redshift stored procedures resulted in the wrong alias name references.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion of Teradata QUALIFY statement.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Improved conversion to carry forward comments to Amazon Redshift and retain a history of changes performed on the view.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Resolved an issue where the RESET WHEN clause didn't result in the correct conversion.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift</td>
<td>Improved conversion of BTEQ scripts that contain MERGE statements.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added built-in functions to improve conversion of PERIOD data type fields.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Amazon Redshift</td>
<td>Enhanced transformation data type mapping for TIME data type.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added access to the initial publication of the AWS Schema Conversion Tool CLI Reference manual in PDF format. See AWS Schema Conversion Tool CLI Reference.</td>
</tr>
</tbody>
</table>

**Issues resolved:**

- General improvements.
Release notes for AWS SCT Build 650

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>Updated and enhanced use of extractor agents, including:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A configuration for use with shared storage and a dedicated copying agent.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Exporting and importing data extraction tasks from one project to another.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Support for Azure SQL Data Warehouse (Azure Synapse) as source.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Using native Netezza partitioning.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For more information, see [Using data extraction agents](p. 215).</td>
</tr>
<tr>
<td>All</td>
<td>Amazon RDS PostgreSQL 13</td>
<td>AWS SCT now supports Amazon RDS PostgreSQL 13 as target.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Improved conversion of a result set from an Microsoft SQL Server procedure to an Aurora PostgreSQL target.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Implemented Oracle to Amazon Redshift conversion improvements.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Implemented improvements to converting dynamic SQL statements.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Implemented improvements to SQL UDF conversion.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Clarified message that AWS SCT doesn't support conversion of EXTERNAL TABLES.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Enhanced built-in conversion functions.</td>
</tr>
<tr>
<td>Teradata BTEQ</td>
<td>Amazon Redshift</td>
<td>Improved handling substitution parameters inside BTEQ scripts while using AWS SCT GUI.</td>
</tr>
<tr>
<td>Microsoft SQL Server DW</td>
<td>All</td>
<td>Upgraded the minimum supported JDBC driver version for Microsoft SQL Server, Azure, Azure Synapse.</td>
</tr>
</tbody>
</table>

**Issues resolved:**

- Teradata: Macro conversion additional improvements [RESOLVED]
- Special characters escaped in the target causing SQL errors and re-work needed to place them back [RESOLVED]
- General improvements

## Release notes for AWS SCT Build 649

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server DW</td>
<td>Amazon Redshift</td>
<td>MSSQL to Amazon Redshift conversion improvements to support temporal tables.</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Implemented built-in function enhancements, such as:                                                                                                       Conversion functions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_BINARY_DOUBLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_BINARY_FLOAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_NUMBER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_DATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_TIMESTAMP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_TIMESTAMP_TZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_DSINTERVAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_YMINTERVAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• VALIDATE_CONVERSION</td>
</tr>
<tr>
<td>Oracle DW</td>
<td>Amazon Redshift</td>
<td>Implemented function enhancements for Approximate Query Processing, such as:                                                                                           Aggregate functions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ANY_VALUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• APPROX_COUNT_DISTINCT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• APPROX_COUNT_DISTINCT_DETAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• APPROX_COUNT_DISTINCT_AGG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LISTAGG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TO_APPROX_COUNT_DISTINCT</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Implemented conversion enhancements for Teradata auto sort and distribution key selection. The DB engine automatically selects distribution and sort keys. Introduced a radio button labeled Use Amazon Redshift automatic table tuning to Current projects settings &gt; Optimization strategies &gt; Initial Key Selection Strategy dialog.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Enhanced AWS SCT table loader to ensure AWS SCT loads all tables from Teradata.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Implemented conversion enhancements so that Amazon Redshift supports correlated subquery patterns that include a simple WHERE NOT EXISTS clause.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added support for use of ECHO commands in macros.</td>
</tr>
</tbody>
</table>
TABLE 1. Source and target systems for what’s new, enhanced, or fixed

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What’s new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 LUW</td>
<td>PostgreSQL</td>
<td>Implemented support for DYNAMIC RESULTS SETS conversion, including:</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td>• Cursor clause WITH RETURN/WITH RETURN TO CLIENT</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>• DYNAMIC RESULT SETS routine clause conversion</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Aurora</td>
<td>Implemented support for current Aurora RDS PostgreSQL as target.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>DB2 LUW</td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td>SAP ASE</td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>MariaDB</td>
<td></td>
<td>Implemented support for MariaDB 10.5 as target.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2 LUW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP ASE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MariaDB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td></td>
<td>Added support of the XMLFOREST function for converting from Oracle to Aurora PostgreSQL.</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Issues resolved:**

- General improvements.

**Release notes for AWS SCT Build 648**

TABLE 2. Source and target systems for what’s new, enhanced, or fixed

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What’s new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Aurora PostgreSQL extension pack custom apply mode implemented: operators for numeric/date and text types.</td>
</tr>
<tr>
<td></td>
<td>Amazon</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL-Compatible Edition</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Aurora</td>
<td>Aurora PostgreSQL Lambda Invoke configuration implemented: aws_lambda extension creation; IAM role assignment to the Aurora PostgreSQL cluster.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2 LUW</td>
<td>Aurora</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td></td>
<td>FORALL statement conversion refactoring implemented:</td>
</tr>
</tbody>
</table>
### ORACLE DW 18,19

**Source**: Oracle DW 18,19  
**Target**: Amazon Redshift  
**What's new, enhanced, or fixed**: Oracle to Amazon Redshift conversion improvements implemented: enhanced conversion built-ins. Aggregate LISTAGG; Analytic LISTAGG.

### ORACLE DW 18,19

**Source**: Oracle DW 18,19  
**Target**: Amazon Redshift  
**What's new, enhanced, or fixed**: Oracle to Amazon Redshift conversion improvements implemented: Query new features.

### VERTICA

**Source**: Vertica  
**Target**: Amazon Redshift  
**What's new, enhanced, or fixed**: Vertica to Amazon Redshift conversion improvements implemented: SSL to JDBC connection with SSL=true.

### MICROSOFT SQL SERVER DW

**Source**: Microsoft SQL Server DW  
**Target**: Amazon Redshift  
**What's new, enhanced, or fixed**: Microsoft SQL Server to Redshift conversion improvements: External Tables.

### TERADATA

**Source**: Teradata  
**Target**: Amazon Redshift  
**What's new, enhanced, or fixed**: Teradata to Redshift conversion improvements: INTERVAL data types arithmetic operations.

**Source**: Teradata  
**Target**: Amazon Redshift  
**What's new, enhanced, or fixed**: Teradata to Redshift conversion improvements: Support for lateral column aliases.

### Issues resolved:

- General improvements

---

### Release notes for AWS SCT Build 647

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server</td>
<td>Microsoft SQL Server</td>
<td>RDS now supports Database Mail feature.</td>
</tr>
</tbody>
</table>
| Microsoft SQL Server | MySQL         | Implementing the maximum name of each type of identifier — The maximum length of object names (for example, tables, constraints, columns) in SQL Server is 128 characters. The maximum length of object names in MySQL is 64 characters. To write converted objects to the MySQL database you need to shorten their names. To prevent duplicate names after cutting, you need to add "checksum" of the original object name to the new names. Cut names longer than 64 characters as follows:  

\[
\text{[first N chars]}() + \"\" + \text{[checksum]}() 
\]

\[
\text{[first N chars]} = 64 - 1 - \text{[length of checksum string]} 
\]

For example:

example_of_a_test_schema_with_a_name_length_greater_than_64_characters

example_of_a_test_schema_with_a_name_length_greater_than_64_9703 |
Release notes for AWS SCT Build 646

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>MySQL/Aurora MySQL</td>
<td>Implemented load and conversion of comments on storage objects. For example, processing of comments on Tables, and processing of comments on Table/View columns.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added support for TIME data type conversion.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Conversion improvements — TD_NORMALIZE_OVERLAP implemented.</td>
</tr>
<tr>
<td>Microsoft SQL Server DW</td>
<td>Amazon Redshift</td>
<td>Conversion improvements — SELECT with WITH clause; SELECT without FROM</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>AWS SCT Data Migration Service Assessor (DMSA) — This new feature enables you to evaluate multiple servers and receive a summary report that shows the best target direction for your environment.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>AWS SCT Wizard — Target comparison now shows differences between targets in a single table view.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Tree Filter UI — Redesigned metadata filter handles more complex filtering patterns.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Assessment Report — Redesigned <strong>Warning</strong> section provides a better description and clearer understanding of an issue.</td>
</tr>
</tbody>
</table>

**Issues resolved:**

- General improvements
- Data Extractors — Subtask failed with ConcurrentModificationException [RESOLVED].
- Microsoft SQL Server to MySQL — max identifier lengths [RESOLVED].
### Issues resolved:

- General improvements
- Greenplum: Unable to run conversion due to the error in the log [RESOLVED].
- MSSQL — PostgreSQL: Transformer error when converting LAG function [RESOLVED].
- MSSQL — PostgreSQL: SCOPE_IDENTITY [RESOLVED].
- AWS SCT hanging in DW projects [RESOLVED].
- Need mapping rule to remove additional space on the column name in AWS SCT [RESOLVED].

### Release notes for AWS SCT Build 645

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Provide solution to resolve Teradata non-fully qualified views (view names or non-fully qualified objects within the view).</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added support of ASCII function to compute nodes.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>When AWS SCT spots multi-byte data in a Teradata CHAR defined as CHAR(N), it is converted to VARCHAR(3*N) in Amazon Redshift.</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Provide Teradata CAST conversion between dates and numbers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SELECT Cast('2020-07-17' AS BIGINT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SELECT Cast(20200630 - 19000000 AS DATE)</td>
</tr>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Support conversion of Teradata PERIOD data types into two Amazon Redshift TIMESTAMP columns:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PERIOD(TIMESTAMP)</td>
</tr>
</tbody>
</table>
Teradata | Amazon Redshift | Support conversion of Teradata RANK function with\nRESET \nWHEN clause.

Teradata | Amazon Redshift | Improved support of CAST in explicit data type conversions, and implicit\nCASTs on expressions.

Teradata | Amazon Redshift | Report unsupported correlated subquery patterns. For more\ninformation, see Correlated subqueries in the Amazon Redshift Database\nDeveloper Guide.

none | Amazon Redshift | Improved tables limit support for RA3 node types.

Teradata | Amazon Redshift | Added support for Teradata geospatial data extraction. For more\ninformation, see Querying spatial data in Amazon Redshift in the\nAmazon Redshift Database Developer Guide.

Microsoft SQL Server | PostgreSQL | Added the option, convert_procedures_to_function.

Issues resolved:

- General improvements

Release notes for AWS SCT Build 644

Changes for AWS SCT releases 1.0.643 are merged into AWS SCT 1.0.644 release.

Teradata | Amazon Redshift | Multiple conversion improvements.

- Improved conversions with QUALIFY with the table alias.
- Improved conversions with the IN operator.
- Improved conversion with the LIKE operator.
- Improved conversions with highlighting issues in converted code.
- Improved conversions with unusual order of WHERE, QUALIFY clauses\nin SQL.
- Fixed transformer errors occurred during JOIN() constructions\nconversion of procedure UPD_FT_SVC_TRANS_BH_CBH_IND.
- Improved conversion of macros to stored procedures.

Added special AWS SCT CLI commands that can parse the provided sql/\nbteq scripts and generate a report on the number of syntax structures\nencountered in the source code.

- Count of BTEQ commands
- Count of HANDLERS
- Count of CAST cases
### What's new, enhanced, or fixed

- Count of DML/DDL cases
- Count of DMLs on updatable views

Added an assessment report action item: Teradata columns with custom date formats are not supported in Amazon Redshift.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>PostgreSQL/Aurora/PostgreSQL</td>
<td>Added functionality to save extension pack installation scripts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed severity level for AI 5334.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improved performance of using a record as package variable IMPLEMENTATION.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added XMLAGG aggregation function support</td>
</tr>
<tr>
<td>IBM Db2</td>
<td>PostgreSQL/Aurora/PostgreSQL</td>
<td>Added load and conversion of comments on storage objects implementation.</td>
</tr>
<tr>
<td>MS SQL DW</td>
<td>Amazon Redshift</td>
<td>Conversion improvement: Resolved issue with PATINDEX.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UI improvements:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Save as SQL for source tree implementation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added additional logic to script generation for multiple files.</td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td>UI improvement: Save as SQL for source tree implementation.</td>
</tr>
</tbody>
</table>

### Issues resolved:

- General improvements to conversions between Teradata and Amazon Redshift
- General bug fixing and UI improvements

### Release notes for AWS SCT Build 642

Changes for AWS Schema Conversion Tool release 1.0.642.

**Note**

AWS Schema Conversion Tool (AWS SCT) build 1.0.642 changes are applicable to Windows, Ubuntu, and Fedora. There is no 1.0.642 build for macOS.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SSIS</td>
<td>AWS Glue</td>
<td>Implemented conversion of Microsoft SQL Server Integration Services (SSIS) ETL packages to AWS Glue. For more information, see Converting SSIS to AWS Glue with AWS SCT (p. 189).</td>
</tr>
<tr>
<td>Oracle</td>
<td>MariaDB/SQL MODE=ORACLE/MYSQL/Amazon</td>
<td>Implemented the PL/SQL declaration section in the WITH clause.</td>
</tr>
</tbody>
</table>
Release notes for AWS SCT build 641

Changes for AWS Schema Conversion Tool release 1.0.641.

Note
AWS Schema Conversion Tool (AWS SCT) build 1.0.641 changes are applicable to Windows, Ubuntu, and Fedora. There is no 1.0.641 build for macOS.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aurora MySQL</td>
<td>Aurora PostgreSQL/</td>
<td>Added support for DBMS_SESSION.RESET_PACKAGE and DBMS_SESSION.MODIFY_PACKAGE.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>Amazon Redshift</td>
<td>Enable exporting of SQL scripts from a Vertica database to Amazon Redshift.</td>
</tr>
<tr>
<td></td>
<td>Amazon Redshift</td>
<td></td>
</tr>
<tr>
<td>Vertica</td>
<td>Amazon Redshift</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vertica</td>
<td></td>
</tr>
</tbody>
</table>

Issues resolved:
- Assessment Report UI Enhancement.
- Add the ability to change JVM settings from UI.
- General improvements.

Resolved:
- Various bug fixes.
- Various performance improvements.
Release notes for AWS SCT Build 640

Changes for AWS SCT releases 1.0.633, 1.0.634, 1.0.635, 1.0.636, 1.0.637, 1.0.638, 1.0.639, and 1.0.640 are merged into AWS SCT 1.0.640 release.

Note
AWS SCT build 1.0.640 changes are applicable to Windows, Ubuntu, and Fedora. They don't apply to macOS. You can't install AWS SCT version 1.0.640 or later on Apple macOS. AWS SCT version 1.0.632 was the last version to support installation on Apple macOS.

In the following tables, you can find lists of the features and bug fixes for the AWS Schema Conversion Tool versions that have been combined into release 1.0.640. These tables group features and bug fixes by the source engine.

Topics
- Release 1.0.640 Oracle changes (p. 339)
- Release 1.0.640 Microsoft SQL Server changes (p. 342)
- Release 1.0.640 MySQL Changes (p. 344)
- Release 1.0.640 PostgreSQL changes (p. 345)
- Release 1.0.640 Db2 LUW changes (p. 347)
- Release 1.0.640 Teradata changes (p. 347)
- Release 1.0.640 changes for other engines (p. 348)

Release 1.0.640 Oracle changes

The following table lists build 1.0.640 changes in which Oracle is the source engine.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Implemented SQL code conversion in Java and Pro*C applications.</td>
</tr>
<tr>
<td></td>
<td>Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Improved performance of the following functions when used in a WHERE clause:</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td>• aws_oracle_ext.to_date</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>• aws_oracle_ext.to_char</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aws_oracle_ext.to_number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aws_oracle_ext.sysdate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aws_oracle_ext.sys_context</td>
</tr>
<tr>
<td>Oracle</td>
<td>RDS MariaDB 10.4</td>
<td>Added RDS MariaDB 10.4 support for all online transactional processing (OLTP) vendors.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL/Aurora PostgreSQL</td>
<td>Added support for DBMS_UTILITY.GET_TIME.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added the following emulations:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DBMS_UTILITY.GET_TIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DBMS_UTILITY.FORMAT_CALL_STACK</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What’s new, enhanced, or fixed</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Oracle</td>
<td>MariaDB/MySQL/Aurora MySQL/</td>
<td>Added sharing clause support for TABLE(DATA,EXTENDED DATA), VIEW(DATA,EXTENDED DATA), and SEQUENCE(DATA)</td>
</tr>
<tr>
<td></td>
<td>Microsoft SQL Server Mode=Oracle/PostgreSQL/Aurora PostgreSQL/RDS Oracle</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL/Aurora PostgreSQL/Oracle RDS</td>
<td>The DEFAULT definition of a column can be extended to have the DEFAULT being applied for explicit NULL insertion. The DEFAULT clause has a new ON NULL clause. This new clause instructs the database to assign a specified default column value when an INSERT statement attempts to assign a value that evaluates to NULL.</td>
</tr>
<tr>
<td>Oracle</td>
<td>MariaDB/MariaDB (SQL MODE=ORACLE)</td>
<td>Added support for &quot;Identity Columns,&quot; which automatically increment at the time of insertion.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Upgrade to Amazon Corretto JDK 11 from JDK 8. For more information, including download links, see What is Amazon Corretto 11? in the Amazon Corretto 11 User Guide.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added information to the assessment report about possible inconsistencies in the user’s database.</td>
</tr>
<tr>
<td>Oracle</td>
<td>MariaDB 10.2/MariaDB 10.3/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL</td>
<td>The DEFAULT clause has a new ON NULL clause, which instructs the database to assign a specified default column value when an INSERT statement attempts to assign a value that evaluates to NULL.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Oracle RDS/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL</td>
<td>Added support for IDENTITY columns.</td>
</tr>
<tr>
<td>Oracle</td>
<td>MySQL 8.x</td>
<td>Added support for CHECK constraint.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL/Aurora PostgreSQL</td>
<td>Implemented checking ANYDATA IS NULL/IS NOT NULL using extension pack routine.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented the emulation of the VALUE function used in a query based on the TABLE function of XMLSequence.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added DBMS_LOB support for the following built-in routines:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DBMS_LOB.CREATETEMPORARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DBMS_LOB.FREETEMPORARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DBMS_LOB.APPEND</td>
</tr>
<tr>
<td>All</td>
<td>SQL Server</td>
<td>SQL Server 2019: Added support for new index attribute OPTIMIZE_FOR_SEQUENTIAL_KEY.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server 2017: Added support for Graph Databases Node and Edge table types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server 2016: Added support for TEMPORAL TABLES.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Implemented the ability to override physical partitions with virtual partitions. Data warehouse extractors extract data according to created virtual partitions.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Amazon Redshift</td>
<td>Implemented conversion of cursor attributes in nested blocks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amazon Redshift doesn’t support collections. Related variables are converted as VARCHAR. All collection operations other than assigning one variable to another are rejected, including initiation and collection elements access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented Amazon Redshift distribution style = AUTO.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL/Aurora PostgreSQL</td>
<td>If a nonreserved word in Oracle is reserved in PostgreSQL, then the following is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the word is quoted, it retains its case and stay quoted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If the word is unquoted, it is cast to uppercase and quoted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented the ability to use functions as input to LTRIM, RTRIM, and TRIM functions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELECT DISTINCT, ORDER BY expressions must appear in select list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For cursor parameters that follow after a parameter with a DEFAULT value, AWS SCT adds DEFAULT IS NULL clause.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Source OUT cursor parameters are converted to IN cursor parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reimplemented package variable by adding the &quot;Package variables logic implementation&quot; option under &quot;Conversion settings'. Available settings are: &quot;session variables&quot; and &quot;plv8 global objects&quot;. The default is &quot;session variables&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented AUTONOMOUS_TRANSACTION pragma support with dblink and pg_background.</td>
</tr>
</tbody>
</table>
Release 1.0.640 Microsoft SQL Server changes

The following table lists build 1.0.640 changes in which Microsoft SQL Server is the source engine.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Azure/</td>
<td>PostgreSQL/Aurora/MySQL/PostgreSQL</td>
<td>Added support for COLUMN STORE indexes.</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDS</td>
<td>MariaDB/MySQL/Aurora/MySQL/PostgreSQL</td>
<td>Added RDS MariaDB 10.4 support for all online transactional processing (OLTP) vendors.</td>
</tr>
<tr>
<td>Azure/SQL Server</td>
<td>MariaDB/MySQL/Aurora/MySQL/PostgreSQL</td>
<td>Added support for the OPTIMIZE_FOR_SEQUENTIAL_KEY index attribute.</td>
</tr>
<tr>
<td>Azure/SQL Server</td>
<td>MySQL/Aurora/MySQL/PostgreSQL</td>
<td>Added support for Databases Node and Edge table types.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Azure/SQL Server</td>
<td>MariaDB/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL</td>
<td>Added support for TEMPORAL TABLES.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Upgrade to Amazon Corretto JDK 11 from JDK 8. For more information, including download links, see <a href="https://aws.amazon.com/corretto/">What is Amazon Corretto 11?</a> in the <em>Amazon Corretto 11 User Guide</em>.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added information to the assessment report about possible inconsistencies in the user's database.</td>
</tr>
<tr>
<td>Azure/SQL Server</td>
<td>MySQL/Aurora MySQL/PostgreSQL/MariaDB</td>
<td>Added support for DML processing for SQL Server Graph Architecture.</td>
</tr>
<tr>
<td>SQL Server</td>
<td>Aurora PostgreSQL</td>
<td>Added option to convert parameters without the <code>par_</code> prefix.</td>
</tr>
<tr>
<td>Azure/SQL Server</td>
<td>MySQL 8.x</td>
<td>Added support for CHECK constraint.</td>
</tr>
<tr>
<td>All</td>
<td>SQL Server</td>
<td>SQL Server 2019: Added support for new index attribute <code>OPTIMIZE_FOR_SEQUENTIAL_KEY</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server 2017: Added support for Graph Databases Node and Edge table types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server 2016: Added support for TEMPORAL TABLES.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Implemented the ability to override physical partitions with virtual partitions. Data warehouse extractors extract data according to created virtual partitions.</td>
</tr>
<tr>
<td>SQL Server</td>
<td>AWS Glue (Python shell)</td>
<td>Conversion improvements, including:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Implemented built-in functions conversion to Python.String.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Implemented EXECUTE and EXEC in stored code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Implemented using table types.</td>
</tr>
<tr>
<td>Azure/SQL Server</td>
<td>PostgreSQL/Aurora PostgreSQL</td>
<td>Implemented making <code>$TMP</code> procedures optional.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SQL Server</td>
<td>MySQL/Aurora</td>
<td>Extended arithmetic operations with dates.</td>
</tr>
<tr>
<td></td>
<td>MySQL</td>
<td>Construction emulation <code>TOP (expression) WITH TIES</code>.</td>
</tr>
<tr>
<td></td>
<td>MySQL</td>
<td>After calling procedures with the generated refcursor out, the refcursor now closes.</td>
</tr>
<tr>
<td></td>
<td>MySQL</td>
<td>Setting a GLOBAL isolation level isn't supported in Aurora MySQL. Only the session scope can</td>
</tr>
<tr>
<td></td>
<td>MySQL</td>
<td>be changed. The default behavior of transactions is to use REPEATABLE READ and consistent reads.</td>
</tr>
<tr>
<td>SQL Server</td>
<td>AWS Glue (Python</td>
<td>SQL Server statements produce a complete result set, but there are times when the results are</td>
</tr>
<tr>
<td></td>
<td>shell)</td>
<td>best processed one row at a time. Opening a cursor on a result set allows processing the result</td>
</tr>
<tr>
<td></td>
<td></td>
<td>set one row at a time. You can assign a cursor to a variable or parameter with a cursor data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented enclosing a series of Transact-SQL statements for stored code so that a group of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transact-SQL statements can be run even though Python doesn't support SQL Server's BEGIN and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END as control-of-flow.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server LABEL and GOTO statements aren't supported by AWS Glue. If AWS SCT encounters a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>label in the code, it is skipped. If AWS SCT encounters a GOTO statement, it is commented.</td>
</tr>
<tr>
<td>SQL Server</td>
<td>Amazon Redshift</td>
<td>Implemented conditional processing of Transact-SQL statements for stored code by implementing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the IF ... ELSE control.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented enclosing a series of Transact-SQL statements for stored code so that a group of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transact-SQL statements can be run as a block. Supports nested BEGIN ... END blocks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented SET and SELECT in stored code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implemented CREATE INDEX in Amazon Redshift (which doesn't support indexes) by creating a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user-specified sort key on the tables.</td>
</tr>
</tbody>
</table>

### Release 1.0.640 MySQL Changes

The following table lists build 1.0.640 changes in which MySQL is the source engine.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What’s new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>PostgreSQL 12.x</td>
<td>Added support for generated columns.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Upgrade to Amazon Corretto JDK 11 from JDK 8. For more information, including download links, see What is Amazon Corretto 11? in the Amazon Corretto 11 User Guide.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added information to the assessment report about possible inconsistencies in the user’s database.</td>
</tr>
</tbody>
</table>
### Release 1.0.640 PostgreSQL changes

The following table lists build 1.0.640 changes in which PostgreSQL is the source engine.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>PostgreSQL/Aurora PostgreSQL 11</td>
<td>Added support for the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Embedded transactions inside SQL stored procedures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The ability to CALL SQL stored procedures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The ability to create SQL stored procedures.</td>
</tr>
<tr>
<td>All</td>
<td>SQL Server</td>
<td>SQL Server 2019: Added support for new index attribute OPTIMIZE_FOR_SEQUENTIAL_KEY.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server 2017: Added support for Graph Databases Node and Edge table types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server 2016: Added support for TEMPORAL TABLES.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Implemented the ability to override physical partitions with virtual partitions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data warehouse extractors extract data according to created virtual partitions.</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>MySQL 8.x</td>
<td>MySQL now supports creation of functional index key parts that index expression values rather than column values. Functional key parts enable indexing of values, such as JSON values, that can't be indexed otherwise. MySQL now supports Now CTE and Recursive CTE.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Upgrade to Amazon Corretto JDK 11 from JDK 8. For more information, including download links, see What is Amazon Corretto 11? in the Amazon Corretto 11 User Guide.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added information to the assessment report about possible inconsistencies in the user's database.</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>PostgreSQL/Aurora PostgreSQL 11</td>
<td>Added support for the following:</td>
</tr>
<tr>
<td>11.x</td>
<td></td>
<td>• Embedded transactions inside SQL stored procedures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The ability to CALL SQL stored procedures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The ability to create SQL stored procedures.</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>MySQL 8.x</td>
<td>Added MySQL support for descending indexes. DESC in an index definition is no longer ignored, but causes storage of key values in descending order. Added MySQL support the use of expressions as default values in data type specifications, including expressions as default values for the BLOB, TEXT, GEOMETRY, and JSON data types. Several existing aggregate functions can now be used as window functions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AVG( )</td>
</tr>
</tbody>
</table>
### What's new, enhanced, or fixed

- BIT_AND()
- BIT_OR()
- BIT_XOR()
- COUNT()
- JSON_ARRAYAGG()
- JSON_OBJECTAGG()
- MAX()
- MIN()
- STDDEV_POP()
- STDDEV()
- STD()
- STDDEV_SAMP()
- SUM()
- VAR_POP()
- VARIANCE()
- VAR_SAMP()

MySQL supports window functions that, for each row from a query, perform a calculation using rows related to that row.

- CUME_DIST()
- DENSE_RANK()
- FIRST_VALUE()
- LAG()
- LAST_VALUE()
- LEAD()
- NTH_VALUE()
- NTILE()
- PERCENT_RANK()
- RANK()
- ROW_NUMBER()

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What’s new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL</td>
<td>MySQL 8.x</td>
<td>Added support for CHECK constraint.</td>
</tr>
<tr>
<td>All</td>
<td>SQL Server</td>
<td>SQL Server 2019: Added support for new index attribute OPTIMIZE_FOR_SEQUENTIAL_KEY. SQL Server 2017: Added support for Graph Databases Node and Edge table types. SQL Server 2016: Added support for TEMPORAL TABLES.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Implemented the ability to override physical partitions with virtual partitions. Data warehouse extractors extract data according to created virtual partitions.</td>
</tr>
</tbody>
</table>
Release 1.0.640 Db2 LUW changes

The following table lists build 1.0.640 changes in which DB2 LUW is the source engine.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PostgreSQL/ Aurora PostgreSQL</td>
<td>All</td>
<td>Added system view sysindexes emulation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If there is a SELECT statement in a procedure without specifying INTO, the parameter INOUT p_refcur of type refcursor is created for a procedure on the target.</td>
</tr>
</tbody>
</table>

Release 1.0.640 Teradata changes

The following table lists build 1.0.640 changes Teradata source engines.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata</td>
<td>Amazon Redshift</td>
<td>Added support for the MERGE and QUALIFY statements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed LOCKING ROWS FOR ACCESS clause from Teradata statements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added support for CAST function.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Upgrade to Amazon Corretto JDK 11 from JDK 8. For more information, including download links, see What Amazon Corretto 11 in the Amazon Corretto 11 User Guide.</td>
</tr>
</tbody>
</table>
Release 1.0.640 changes for other engines

The following table lists build 1.0.640 changes for other source engines.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What’s new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sybase</td>
<td>RDS</td>
<td>Added RDS MariaDB 10.4 support for all online transactional processing (OLTP) vendors.</td>
</tr>
<tr>
<td></td>
<td>MariaDB 10.4</td>
<td></td>
</tr>
<tr>
<td>SAP ASE</td>
<td>MariaDB</td>
<td>Implemented the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MariaDB 10.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• EXECUTE IMMEDIATE statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DEFAULT definitions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CHECK constraint support</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>PostgreSQL 12.x</td>
<td>Added support for generated columns.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Upgrade to Amazon Corretto JDK 11 from JDK 8. For more information, including download links, see What is Amazon Corretto 11? in the Amazon Corretto 11 User Guide.</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>Added information to the assessment report about possible inconsistencies in the user’s database.</td>
</tr>
<tr>
<td>Source</td>
<td>Target</td>
<td>What's new, enhanced, or fixed</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>MySQL 8.0.17</td>
<td>Added CHECK constraint support.</td>
</tr>
</tbody>
</table>
| All | SQL Server | SQL Server 2019: Added support for new index attribute OPTIMIZE_FOR_SEQUENTIAL_KEY.  
SQL Server 2017: Added support for Graph Databases Node and Edge table types.  
SQL Server 2016: Added support for TEMPORAL TABLES. |
| Vertica | Amazon Redshift | Added support for distribution style = AUTO. |
| All | All | Implemented the ability to override physical partitions with virtual partitions. Data warehouse extractors extract data according to created virtual partitions. |
| Amazon Redshift | Amazon Redshift | Unsupported built-in functions in DML statements are replaced with NULL as a placeholder. |
| Sybase | PostgreSQL | Added support for native functions. |
| SAP ASE | MySQL/Aurora MySQL | The default isolation level for Aurora MySQL is REPEATABLE READ. Setting a GLOBAL isolation level isn't supported in Aurora MySQL. Only session scope can be changed. The default behavior of transactions is to use REPEATABLE READ and consistent reads. Applications designed to run with READ COMMITTED may need to be modified. Or you can explicitly change the default to READ COMMITTED. |
| SAP ASE | PostgreSQL | Added support for the CONVERT function(optimistic) without the extension pack. |
| SAP ASE | All | Added system view sysindexes emulation.  
If there is a SELECT statement in a procedure without specifying INTO, the parameter INOUT p_refcur of type refcursor is created for a procedure on the target. |
| Greenplum | Amazon Redshift | Implemented CREATE TEMPORARY TABLE as follows:  
Greenplum temporary tables have parameters GLOBAL and LOCAL. These keywords are present for SQL standard compatibility, but have no effect in a Greenplum Database.  
The new table is created as a temporary table. Temporary tables are automatically dropped at the end of a session, or optionally at the end of the current transaction (see ON COMMIT). Existing permanent tables with the same name aren't visible to the current session while the temporary table exists, unless they are referenced with schema-qualified names. Any indexes created on a temporary table are automatically temporary as well. |
# Document history

The following table describes the important changes to the AWS Schema Conversion Tool (AWS SCT) user guide after January 2018.

You can subscribe to an RSS feed to be notified of updates to this documentation.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS SCT build #1.0.662</td>
<td>Build 1.0.662 implements SQL code conversion in C# applications and improves the multiserver assessment report workflow. It also adds multiple conversion improvements and resolves a number of conversion issues.</td>
<td>May 19, 2022</td>
</tr>
<tr>
<td>AWS SCT build #1.0.661</td>
<td>Build 1.0.661 provides support of IBM Db2 for z/OS as a migration source. It also adds support for conversion of extract, transform, and load (ETL) scripts to AWS Glue Studio and resolves a number of conversion issues.</td>
<td>April 21, 2022</td>
</tr>
<tr>
<td>AWS SCT build #1.0.660</td>
<td>Build 1.0.660 provides support of PostgreSQL major version 14 and MariaDB 10.6 as migration targets. It also adds support for conversion of Oracle indexes for materialized views, and resolves a number of conversion issues.</td>
<td>March 21, 2022</td>
</tr>
<tr>
<td>AWS SCT build #1.0.659</td>
<td>Build 1.0.659 provides support of PostgreSQL major version 13 on Aurora PostgreSQL-Compatible Edition as a migration target. It implements SQL code conversion in C# applications, adds support of Oracle Unified Auditing, and resolves a number of conversion issues.</td>
<td>February 21, 2022</td>
</tr>
<tr>
<td>AWS SCT build #1.0.658</td>
<td>Build 1.0.658 provides integration with AWS Secrets Manager and adds support of Amazon Redshift virtual target database platform. It also adds a number of conversion improvements and bug fixes.</td>
<td>January 20, 2022</td>
</tr>
<tr>
<td>AWS SCT build #1.0.657</td>
<td>Build 1.0.657 improves conversion from Microsoft SQL</td>
<td>December 20, 2021</td>
</tr>
</tbody>
</table>
Server to Aurora PostgreSQL-Compatible Edition, Amazon RDS for PostgreSQL, and other migration destinations. It also adds a number of user interface improvements and bug fixes.

<p>| AWS SCT build #1.0.656 | Build 1.0.656 provides support of multiple source and target databases in one project. It also adds conversion, optimization strategy, and general improvements and a number of bug fixes. | November 22, 2021 |
| AWS SCT build #1.0.655 | Build 1.0.655 implements conversion of Teradata FastExport job scripts to Amazon Redshift RSQL and increases the minimum supported driver version for Greenplum to 42.2.5. It also adds a number of improvements and bug fixes. | October 18, 2021 |
| AWS SCT build #1.0.654 | Build 1.0.654 implements conversion of Shell, Teradata FastLoad, and Teradata Basic Teradata Query (BTEQ) scripts to Amazon Redshift RSQL. It also resolves a number of conversion issues and adds a number of improvements and bug fixes. | September 16, 2021 |
| AWS SCT build #1.0.653 | Build 1.0.653 implements conversion of dynamic SQL created in called functions or procedures. It also improves conversion of encrypted routines and adds a number of improvements and bug fixes. | August 10, 2021 |
| AWS SCT build #1.0.652 | Build 1.0.652 implements script command mode in the command-line interface and implements schema optimization rules. It also adds a number of conversion and performance improvements and bug fixes. | June 30, 2021 |
| AWS SCT build #1.0.651 | Build 1.0.651 adds a number of improvements and bug fixes. It also provides access to the initial copy of the AWS Schema Conversion Tool CLI Reference. | June 4, 2021 |</p>
<table>
<thead>
<tr>
<th>AWS SCT build #1.0.650</th>
<th>Build 1.0.650 implements support of Amazon RDS for PostgreSQL 13 as a target database, updates extractor agents. It also upgrades the minimum supported JDBC driver version for Microsoft SQL Server, Azure, and Azure Synapse. In addition, it adds a number of conversion improvements and bug fixes.</th>
<th>April 30, 2021</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS SCT build #1.0.649</td>
<td>Build 1.0.649 implements support of MariaDB 10.5 as a target database and implements function enhancements for conversion of Oracle built-in functions. It also adds a number of conversion and performance improvements and bug fixes.</td>
<td>March 29, 2021</td>
</tr>
<tr>
<td>AWS SCT build #1.0.648</td>
<td>Build 1.0.648 adds a number of conversion improvements and bug fixes.</td>
<td>February 22, 2021</td>
</tr>
<tr>
<td>AWS SCT build #1.0.647</td>
<td>Build 1.0.647 adds support of the Database Mail feature on Amazon RDS, implements load and conversion of comments on storage objects. It also adds AWS SCT Data Migration Service Assessor and AWS SCT Wizard and implements the tree filter user interface. In addition, it adds a redesigned section in the Assessment Report and a number of improvements and bug fixes.</td>
<td>January 15, 2021</td>
</tr>
<tr>
<td>AWS SCT build #1.0.646</td>
<td>Build 1.0.646 adds support for INTERVAL data types, Identity columns, and cursors conversion, and adds a number of improvements and bug fixes.</td>
<td>December 28, 2020</td>
</tr>
<tr>
<td>AWS SCT build #1.0.645</td>
<td>Build 1.0.645 adds support for ETL SSIS to AWS Glue conversion and a number of improvements and bug fixes.</td>
<td>November 16, 2020</td>
</tr>
<tr>
<td>AWS SCT build #1.0.643-1.0.644</td>
<td>Build 1.0.644 adds a number of conversion, performance, and user-interface improvements and bug fixes.</td>
<td>October 14, 2020</td>
</tr>
<tr>
<td>Version</td>
<td>Changes</td>
<td>Date</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>AWS SCT build #1.0.642</td>
<td>Build 1.0.642 implements conversion of ETL packages from Microsoft SQL Server Integration Services to AWS Glue and adds a number of improvements and bug fixes.</td>
<td>August 28, 2020</td>
</tr>
<tr>
<td>AWS SCT build #1.0.641</td>
<td>Added SSL support for data extractors. Build also includes a number of improvements and fixes.</td>
<td>July 17, 2020</td>
</tr>
<tr>
<td>AWS SCT builds #1.0.633-1.0.640</td>
<td>Upgraded from JDK 8 to Amazon Corretto JDK 11. Added tables identifying other upgrades, changes, and fixes.</td>
<td>June 22, 2020</td>
</tr>
<tr>
<td>AWS WQF availability</td>
<td>AWS SCT is no longer providing the AWS Workload Qualification Framework (AWS WQF) tool for download.</td>
<td>June 19, 2020</td>
</tr>
<tr>
<td>AWS SCT builds #1.0.632</td>
<td>SCT UI - Added new tab to show errors that happen when applying scripts. You can now save the source tree as SQL when converting from SAP ASE. Improvements for conversions to PostgreSQL or Aurora PostgreSQL or Redshift.</td>
<td>November 19, 2019</td>
</tr>
<tr>
<td>AWS SCT builds #1.0.631 and #1.0.630 (combined)</td>
<td>Better support ROWIDs in Oracle, and for system objects in Microsoft SQL Server and SAP ASE. Better handling for missing specifiers of SQL Server schemas. Better support for conversions from Greenplum to Redshift. Improved support for conversion of stored code when moving to Amazon Redshift, MariaDB, MySQL, and PostgreSQL.</td>
<td>September 30, 2019</td>
</tr>
<tr>
<td>AWS SCT build #1.0.629</td>
<td>Support for stored procedures for conversions from Netezza. Improved support for conversions to Amazon Redshift, DynamoDB, MySQL, and PostgreSQL. Added support for SAP ASE 12.5 as a source.</td>
<td>August 20, 2019</td>
</tr>
<tr>
<td>AWS SCT build #1.0.628</td>
<td>Support for service emulation for conversions from DB2, SQL Server and Oracle. Enhancements for conversions to Amazon Redshift, including more support for cursors and stored procedures.</td>
<td>June 22, 2019</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>AWS SCT build #1.0.627</td>
<td>Support for conversions from SQL Server to stored procedures in Amazon Redshift. Enhancements for converting to PostgreSQL 11 and MySQL 8.0.</td>
<td>May 31, 2019</td>
</tr>
<tr>
<td>AWS SCT build #1.0.626</td>
<td>PostgreSQL 11 and MySQL 8.0 are now supported as targets. SAP ASE 15.5 is now supported as a source.</td>
<td>April 26, 2019</td>
</tr>
<tr>
<td>AWS SCT build #1.0.625</td>
<td>Updates include the ability to convert Teradata BTEQ to AWS Glue, support for conversions to MariaDB 10.3 with Oracle compatibility mode support, support for SAP ASE 15.7, and service substitutions to emulate missing functionality.</td>
<td>March 25, 2019</td>
</tr>
<tr>
<td>AWS SCT build #1.0.624</td>
<td>Updates include the ability to convert Oracle ETL to AWS Glue, and support for conversions from Microsoft SQL Server, Oracle, and IBM Db2 LUW to Amazon RDS for MariaDB. We also added support for conversions from SAP ASE to RDS for MySQL and Amazon Aurora with MySQL compatibility. In addition, we added support for the Orafce extension during Oracle conversion to PostgreSQL.</td>
<td>February 22, 2019</td>
</tr>
<tr>
<td>AWS SCT build #1.0.623</td>
<td>Updates include the ability to convert SAP ASE databases, and the ability to convert T-SQL scripts, DML, and DDL to equivalent code or components. We also added Oracle and Microsoft SQL Server emulations to improve conversions.</td>
<td>January 25, 2019</td>
</tr>
<tr>
<td>AWS SCT build #1.0.622</td>
<td>Updates include the Workload Qualification Framework, which analyzes the workload for an entire migration, including database and app modifications.</td>
<td>December 20, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.621</td>
<td>Updates include support for Aurora PostgreSQL 10 as a target, and the ability to migrate from Netezza using external table options.</td>
<td>November 21, 2018</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>AWS SCT build #1.0.620</td>
<td>Updates include the ability to save SQL scripts, and support for Oracle global cursors when migrating to MySQL.</td>
<td>October 22, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.619</td>
<td>Updates include support for migrating from Apache Cassandra to DynamoDB, and support for Vertica 9 as a source.</td>
<td>September 20, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.618</td>
<td>Updates include expanded assessment reports, support for converting Oracle ROWIDs, and support for SQL Server user-defined tables.</td>
<td>August 24, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.617</td>
<td>Updates include expanded assessment reports, support for converting Oracle ROWIDs, and support for SQL Server user-defined tables.</td>
<td>July 24, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.616</td>
<td>Updates include support for RDS when converting from Oracle to Amazon RDS for Oracle, converting Oracle schedule objects, and support for Oracle jobs, partitioning, and Db2 LUW version 10.1.</td>
<td>June 26, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.615</td>
<td>Updates include support for SQL Server to PostgreSQL GOTO statements, PostgreSQL 10 partitioning, and Db2 LUW version 10.1.</td>
<td>May 24, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.614</td>
<td>Updates include support for Oracle to Oracle DB Links, SQL Server to PostgreSQL inline functions, and emulation of Oracle system objects.</td>
<td>April 25, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.613</td>
<td>Updates include support for Db2 LUW, conversion of SQL*Plus files, and SQL Server Windows Authentication.</td>
<td>March 28, 2018</td>
</tr>
<tr>
<td>AWS SCT build #1.0.612</td>
<td>Updates include support for custom data type mapping, schema compare for Oracle 10, and Oracle to PostgreSQL conversion of global variables.</td>
<td>February 22, 2018</td>
</tr>
</tbody>
</table>
Earlier updates

The following table describes the important changes to the AWS Schema Conversion Tool (AWS SCT) user guide prior to January 2018.

<table>
<thead>
<tr>
<th>Version</th>
<th>Change</th>
<th>Description</th>
<th>Date changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.608</td>
<td>FIPS endpoint support for Amazon S3</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. 34).</td>
<td>November 17, 2017</td>
</tr>
<tr>
<td>1.0.607</td>
<td>FIPS endpoint support for Amazon S3</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. 34).</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. 234).</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. 234).</td>
<td>August 28, 2017</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 filters</td>
<td>You can now filter the data that the extraction agents extract from your data warehouse. For more information, see Creating data migration rules in AWS SCT (p. 230).</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 Teradata 13 and Oracle Data Warehouse 10 schemas to equivalent Amazon Redshift schemas. For more</td>
<td>June 16, 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></td>
<td>warehouse versions</td>
<td>information, see Converting data warehouse schemas to Amazon Redshift using AWS SCT (p. 157).</td>
<td></td>
</tr>
<tr>
<td>1.0.602</td>
<td>Data extraction agents support additional data warehouses</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. 215).</td>
<td>May 11, 2017</td>
</tr>
<tr>
<td>1.0.602</td>
<td>Data extraction agents can copy data to Amazon Redshift</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. 234).</td>
<td>May 11, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>AWS SCT supports additional data warehouses</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 using AWS SCT (p. 157).</td>
<td>April 18, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>Data extraction agents support additional data warehouses</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. 215).</td>
<td>April 18, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>Data extraction agents support additional operating systems</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. 224).</td>
<td>April 18, 2017</td>
</tr>
<tr>
<td>1.0.601</td>
<td>Data extraction agents upload to Amazon S3 automatically</td>
<td>Data extraction agents now upload your extracted data to Amazon S3 automatically. For more information, see Data extraction task output (p. 243).</td>
<td>April 18, 2017</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. 215).</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 feedback (p. 4).</td>
<td>February 16, 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>1.0.502</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 AWS SCT with AWS DMS (p. 213).</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 AWS SCT (p. 142).</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 Storing AWS service profiles in the AWS SCT (p. 34).</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 using AWS SCT (p. 157).</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 AWS SCT (p. 179).</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 migration rules in AWS SCT (p. 160).</td>
<td>October 4, 2016</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 migration assessment reports with AWS SCT (p. 125).</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 using AWS SCT (p. 157).</td>
<td>July 13, 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>1.0.400</td>
<td>Application SQL</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 AWS SCT (p. 268).</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. 298) and Using the custom Python library for AWS SCT extension packs (p. 297).</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>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 Create New Database Migration Project 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>