# Table of Contents

- What Is the AWS Schema Conversion Tool? .................................................................................................................. 1
  - Schema Conversion Overview ................................................................................................................................. 2
- Feedback ........................................................................................................................................................................... 3
- Installing, Verifying, and Updating .............................................................................................................................. 4
  - Installing the AWS SCT ........................................................................................................................................... 4
    - Installing Previous Versions ................................................................................................................................. 5
  - Verifying the AWS SCT File Download .................................................................................................................... 5
    - Verifying the Checksum of the AWS SCT File .................................................................................................. 6
    - Verifying the AWS SCT RPM Files on Fedora ................................................................................................. 6
    - Verifying the AWS SCT DEB Files on Ubuntu ................................................................................................. 7
    - Verifying the AWS SCT MSI File on Microsoft Windows ............................................................................ 7
    - Verifying the AWS SCT Application on Mac OS ............................................................................................ 7
- Installing the Required Database Drivers ..................................................................................................................... 8
  - Installing JDBC Drivers on Linux .......................................................................................................................... 9
  - Storing Driver Paths in the Global Settings ........................................................................................................... 10
- Updating the AWS SCT .................................................................................................................................................. 11
- Using the AWS SCT User Interface ............................................................................................................................ 12
  - The AWS SCT Project Window ............................................................................................................................. 12
  - Storing AWS Profiles ............................................................................................................................................ 13
    - Storing AWS Credentials .................................................................................................................................. 14
    - Setting the Default Profile for a Project ........................................................................................................... 16
  - Storing Database Passwords ................................................................................................................................. 16
  - Using the Union All View for Projects with Partitioned Tables ............................................................................. 16
  - AWS SCT Tree Filters ............................................................................................................................................. 16
  - Importing a File List for the Tree Filter ................................................................................................................ 17
  - Hiding Schemas ....................................................................................................................................................... 18
  - Keyboard Shortcuts .................................................................................................................................................. 19
  - The Database Migration Assessment Report ...................................................................................................... 20
  - Starting SCT ........................................................................................................................................................... 24
  - Creating a Project .................................................................................................................................................... 25
  - Converting Your Schema ........................................................................................................................................ 26
  - Applying the Converted Schema to Your Target DB Instance ........................................................................... 27
- Getting Started .............................................................................................................................................................. 29
  - Source Databases for AWS SCT .......................................................................................................................... 30
    - Using Oracle as a Source for AWS Schema Conversion Tool ........................................................................ 30
      - Permissions Needed for Oracle as a Source ................................................................................................. 31
      - Connecting to Oracle as a Source ................................................................................................................ 31
      - Oracle to PostgreSQL .................................................................................................................................... 33
      - Oracle to MySQL ............................................................................................................................................. 36
      - Oracle to Amazon RDS Oracle ...................................................................................................................... 40
    - Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool ...................................................... 43
      - Source: Microsoft SQL Server ....................................................................................................................... 43
      - Source: Windows Authentication with Microsoft SQL Server .................................................................. 43
      - Connecting to SQL Server as a Source ........................................................................................................ 45
      - SQL Server to MySQL .................................................................................................................................. 46
      - SQL Server to PostgreSQL .......................................................................................................................... 48
      - SQL Server to Amazon RDS SQL Server ................................................................................................. 51
    - Using MySQL as a Source for AWS Schema Conversion Tool .............................................................................. 52
      - Source: MySQL .............................................................................................................................................. 52
      - Connecting to MySQL as a Source .............................................................................................................. 52
    - Using PostgreSQL as a Source for AWS Schema Conversion Tool ...................................................................... 54
      - Source: PostgreSQL .................................................................................................................................... 54
      - Connecting to PostgreSQL as a Source ..................................................................................................... 55

Version 1.0
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Db2 LUW as a Source for AWS Schema Conversion Tool</td>
<td>56</td>
</tr>
<tr>
<td>Source: Db2 LUW</td>
<td>56</td>
</tr>
<tr>
<td>Connecting to Db2 LUW as a Source</td>
<td>58</td>
</tr>
<tr>
<td>DB2LUW to PostgreSQL</td>
<td>60</td>
</tr>
<tr>
<td>Using Amazon Redshift as a Source for AWS Schema Conversion Tool</td>
<td>61</td>
</tr>
<tr>
<td>Source: Amazon Redshift</td>
<td>61</td>
</tr>
<tr>
<td>Connecting to Redshift as a Source</td>
<td>61</td>
</tr>
<tr>
<td>Using Oracle DW as a Source for AWS Schema Conversion Tool</td>
<td>63</td>
</tr>
<tr>
<td>Source: Oracle Data Warehouse</td>
<td>63</td>
</tr>
<tr>
<td>Connecting to OracleDW as a Source</td>
<td>63</td>
</tr>
<tr>
<td>Using Teradata as a Source for AWS Schema Conversion Tool</td>
<td>66</td>
</tr>
<tr>
<td>Source: Teradata</td>
<td>66</td>
</tr>
<tr>
<td>Connecting to Teradata as a Source</td>
<td>66</td>
</tr>
<tr>
<td>Using Netezza as a Source for AWS Schema Conversion Tool</td>
<td>69</td>
</tr>
<tr>
<td>Source: Netezza</td>
<td>69</td>
</tr>
<tr>
<td>Connecting to Netezza as a Source</td>
<td>69</td>
</tr>
<tr>
<td>Using Greenplum as a Source for AWS Schema Conversion Tool</td>
<td>71</td>
</tr>
<tr>
<td>Source: Greenplum</td>
<td>71</td>
</tr>
<tr>
<td>Connecting to Greenplum as a Source</td>
<td>71</td>
</tr>
<tr>
<td>Using Vertica as a Source for AWS Schema Conversion Tool</td>
<td>72</td>
</tr>
<tr>
<td>Source: Vertica</td>
<td>73</td>
</tr>
<tr>
<td>Connecting to Vertica as a Source</td>
<td>73</td>
</tr>
<tr>
<td>Using Microsoft SQL Server DW as a Source for AWS Schema Conversion Tool</td>
<td>74</td>
</tr>
<tr>
<td>Source: Microsoft SQL Server Data Warehouse</td>
<td>75</td>
</tr>
<tr>
<td>Connecting to SQLServerDW as a Source</td>
<td>75</td>
</tr>
<tr>
<td>Creating Conversion Reports</td>
<td>77</td>
</tr>
<tr>
<td>Migration Assessment Reports</td>
<td>77</td>
</tr>
<tr>
<td>Creating a Database Migration Assessment Report</td>
<td>78</td>
</tr>
<tr>
<td>Viewing the Assessment Report</td>
<td>79</td>
</tr>
<tr>
<td>Saving the Assessment Report</td>
<td>80</td>
</tr>
<tr>
<td>WQF Migration Reports</td>
<td>81</td>
</tr>
<tr>
<td>Setting Up AWS WQF</td>
<td>82</td>
</tr>
<tr>
<td>Preparing for Batch Imports</td>
<td>83</td>
</tr>
<tr>
<td>Creating Reports</td>
<td>85</td>
</tr>
<tr>
<td>Understanding Workload Categories</td>
<td>89</td>
</tr>
<tr>
<td>Converting Database Schemas</td>
<td>92</td>
</tr>
<tr>
<td>Creating Mapping Rules</td>
<td>93</td>
</tr>
<tr>
<td>Creating Mapping Rules</td>
<td>94</td>
</tr>
<tr>
<td>Creating Mapping Rules for Objects</td>
<td>95</td>
</tr>
<tr>
<td>Exporting Mapping Rules</td>
<td>95</td>
</tr>
<tr>
<td>Converting Your Schema</td>
<td>95</td>
</tr>
<tr>
<td>Converting Schema</td>
<td>95</td>
</tr>
<tr>
<td>Editing Converted Schema</td>
<td>97</td>
</tr>
<tr>
<td>Clearing a Converted Schema</td>
<td>98</td>
</tr>
<tr>
<td>Handling Manual Conversions</td>
<td>99</td>
</tr>
<tr>
<td>Modifying Your Source Schema</td>
<td>99</td>
</tr>
<tr>
<td>Modifying Your Target Schema</td>
<td>100</td>
</tr>
<tr>
<td>Updating and Refreshing Your Converted Schema</td>
<td>100</td>
</tr>
<tr>
<td>Saving and Applying Your Schema</td>
<td>101</td>
</tr>
<tr>
<td>Saving Your Converted Schema to a File</td>
<td>101</td>
</tr>
<tr>
<td>Applying Your Converted Schema</td>
<td>102</td>
</tr>
<tr>
<td>The Extension Pack Schema</td>
<td>103</td>
</tr>
<tr>
<td>Comparing Schemas</td>
<td>104</td>
</tr>
<tr>
<td>Related Transformed Objects</td>
<td>105</td>
</tr>
<tr>
<td>Converting Data Warehouse Schemas to Amazon Redshift</td>
<td>106</td>
</tr>
<tr>
<td>Choosing Optimization Strategies and Rules</td>
<td>107</td>
</tr>
<tr>
<td>Collecting or Uploading Statistics</td>
<td>108</td>
</tr>
<tr>
<td>Page</td>
<td>Topic</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>111</td>
<td>Overview of Converting Application SQL</td>
</tr>
<tr>
<td>123</td>
<td>Converting ETL Processes to AWS Glue</td>
</tr>
<tr>
<td>134</td>
<td>Converting Using the Python API for AWS Glue</td>
</tr>
<tr>
<td>135</td>
<td>Converting Using AWS Glue in the AWS SCT UI</td>
</tr>
<tr>
<td>138</td>
<td>Limitations</td>
</tr>
<tr>
<td>124</td>
<td>AWS Glue Data Catalog</td>
</tr>
<tr>
<td>125</td>
<td>Prerequisites</td>
</tr>
<tr>
<td>126</td>
<td>Converting Your Schema</td>
</tr>
<tr>
<td>127</td>
<td>Step 1: Create a New Project</td>
</tr>
<tr>
<td>127</td>
<td>Step 2: Create an AWS Glue Job</td>
</tr>
<tr>
<td>130</td>
<td>Converting Using the Python API for AWS Glue</td>
</tr>
<tr>
<td>131</td>
<td>Step 1: Create a Database</td>
</tr>
<tr>
<td>132</td>
<td>Step 2: Create a Connection</td>
</tr>
<tr>
<td>133</td>
<td>Step 3: Creating an AWS Glue Crawler</td>
</tr>
<tr>
<td>134</td>
<td>Using the AWS Schema Conversion Tool with the AWS Database Migration Service</td>
</tr>
<tr>
<td>134</td>
<td>Using an AWS SCT Replication Agent with AWS DMS</td>
</tr>
<tr>
<td>134</td>
<td>Using an AWS SCT Data Extraction Agent with AWS DMS</td>
</tr>
<tr>
<td>135</td>
<td>Migrating Data From an On-Premises Data Warehouse to Amazon Redshift</td>
</tr>
<tr>
<td>137</td>
<td>Prerequisite Settings</td>
</tr>
<tr>
<td>138</td>
<td>Installing Agents</td>
</tr>
<tr>
<td>141</td>
<td>Registering Agents</td>
</tr>
<tr>
<td>142</td>
<td>Hiding and Recovering Information for an AWS SCT Agent</td>
</tr>
<tr>
<td>143</td>
<td>Creating Data Extraction Filters</td>
</tr>
<tr>
<td>144</td>
<td>Sorting Data</td>
</tr>
<tr>
<td>145</td>
<td>Creating, Running, and Monitoring an AWS SCT Task</td>
</tr>
<tr>
<td>147</td>
<td>Data Extraction Using a AWS Snowball Device</td>
</tr>
<tr>
<td>153</td>
<td>Data Extraction Task Output</td>
</tr>
<tr>
<td>154</td>
<td>Using Virtual Partitioning</td>
</tr>
<tr>
<td>157</td>
<td>Working with LOBs</td>
</tr>
<tr>
<td>158</td>
<td>Best Practices and Troubleshooting</td>
</tr>
<tr>
<td>159</td>
<td>Migrating Data From Apache Cassandra to Amazon DynamoDB</td>
</tr>
<tr>
<td>161</td>
<td>Prerequisites for Migrating From Cassandra to DynamoDB</td>
</tr>
<tr>
<td>163</td>
<td>Create a New AWS SCT Project</td>
</tr>
<tr>
<td>165</td>
<td>Create a Clone Datacenter</td>
</tr>
<tr>
<td>172</td>
<td>Install, Configure and Run the Data Extraction Agent</td>
</tr>
<tr>
<td>175</td>
<td>Migrate Data From the Clone Datacenter to Amazon DynamoDB</td>
</tr>
<tr>
<td>179</td>
<td>Post-Migration Activities</td>
</tr>
<tr>
<td>180</td>
<td>Converting Application SQL</td>
</tr>
<tr>
<td>180</td>
<td>Overview of Converting Application SQL</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 DB instance, an Amazon Aurora DB cluster, an Amazon RDS PostgreSQL DB instance, or an Amazon Redshift cluster. The converted schema can also be used with a database on an Amazon EC2 instance or stored as data on an Amazon S3 bucket.

AWS SCT supports several industry standards, including Federal Information Processing Standards (FIPS), 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 on Amazon RDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server (version 2008 and later)</td>
<td>Amazon Aurora with MySQL compatibility, Amazon Aurora with PostgreSQL compatibility, MariaDB 10.2 and 10.3, Microsoft SQL Server, MySQL, PostgreSQL</td>
</tr>
<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 Migrating Data to an Amazon Aurora DB Cluster.</td>
</tr>
<tr>
<td>Oracle (version 10.2 and later)</td>
<td>Aurora MySQL, Aurora PostgreSQL, MariaDB 10.2 and 10.3, MySQL, Oracle, PostgreSQL</td>
</tr>
<tr>
<td>PostgreSQL (version 9.1 and later)</td>
<td>Aurora MySQL, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>IBM Db2 LUW (versions 9.1, 9.5, 9.7, 10.5, and 11.1)</td>
<td>Aurora MySQL, MariaDB 10.2 and 10.3, MySQL, PostgreSQL, Aurora PostgreSQL</td>
</tr>
<tr>
<td>Apache Cassandra (versions 2.0, 3.0, 3.1.1, and 3.11.2)</td>
<td>Amazon DynamoDB</td>
</tr>
<tr>
<td>Sybase (16.0 and 15.7)</td>
<td>Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL</td>
</tr>
</tbody>
</table>

AWS SCT supports the following data warehouse conversions.

<table>
<thead>
<tr>
<th>Source Database</th>
<th>Target Database on Amazon Redshift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greenplum Database (version 4.3 and later)</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>Microsoft SQL Server (version 2008 and later)</td>
<td>Amazon Redshift</td>
</tr>
</tbody>
</table>
Schema Conversion Overview

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

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

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

For information on the conversion process, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 92).

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. 135).

- You can use AWS SCT to create AWS DMS endpoints and tasks. You can run and monitor these tasks from AWS SCT. For more information, see Using the AWS Schema Conversion Tool with the AWS Database Migration Service (p. 134).

- In some cases, database features can't be converted to equivalent Amazon RDS or Amazon Redshift features. The AWS SCT extension pack wizard can help you install AWS Lambda functions and Python libraries to emulate the features that can't be converted. For more information, see Using the AWS Schema Conversion Tool Extension Pack (p. 186).

- You can use AWS SCT to optimize your existing Amazon Redshift database. AWS SCT recommends sort keys and distribution keys to optimize your database. For more information, see Optimizing Amazon Redshift by Using the AWS Schema Conversion Tool (p. 121).

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

- You can use AWS SCT to convert SQL in your C++, C#, Java, or other application code. You can view, analyze, edit, and save the converted SQL code. For more information, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 180).
Providing Customer Feedback

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

To provide feedback about AWS SCT.

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

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

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

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

Installing the AWS SCT

To install the AWS SCT

1. Download the compressed file that contains the AWS SCT installer, using the link for your operating system. All compressed files have a .zip extension. When you extract the AWS SCT installer file, it will be in the appropriate format for your operating system.
   - Microsoft Windows
   - Apple macOS
   - Ubuntu Linux (.deb)
   - Fedora Linux (.rpm)

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

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

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

<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 the AWS SCT

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

https://d211wdulfroga6.cloudfront.net/builds/1.0/<version>/<OS>/aws-schema-conversion-tool-1.0.zip

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

- MacOS - https://d211wdulfroga6.cloudfront.net/builds/1.0/607/MacOS/aws-schema-conversion-tool-1.0.zip
- Windows - https://d211wdulfroga6.cloudfront.net/builds/1.0/607/Windows/aws-schema-conversion-tool-1.0.zip
- Ubuntu - https://d211wdulfroga6.cloudfront.net/builds/1.0/607/Ubuntu/aws-schema-conversion-tool-1.0.zip
- Fedora - https://d211wdulfroga6.cloudfront.net/builds/1.0/607/Fedora/aws-schema-conversion-tool-1.0.zip

Verifying the AWS SCT File Download

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

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

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

To verify the AWS SCT distribution file using a checksum

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

   Fedora   b4f5f66f91bfc1c312e2827e960691c269a9002cd1371cf1841593f88cbb5e6
   Ubuntu   4315eb6664494d4fc395932351f00399adb6c6c64b9f30adda2eecc903c54eca4
   Windows  6e29679a4c353c5396a05d8d50f308981e4ec34bd0a608847470700a0ae9a23
   MacOs    ed56d3a4b9309e92a4d2ef439d35449ac1326f470c23cd5866e1b0a60b0e67

3. Run the SHA256 validation command for your operating system in the directory that contains the distribution file. For example, the command to run on the Mac operating system is the following:

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

4. Compare the results of the command with the value shown in the sha256Check.txt file. The two values should match.

Verifying the AWS SCT RPM Files on Fedora

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

To verify the AWS SCT RPM files on Fedora

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

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

6. Check that the import was successful by running the following command:
Verifying the AWS SCT DEB Files on Ubuntu

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

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

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

Verifying the AWS SCT Application on Mac OS

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

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

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

Installing the Required Database Drivers

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

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

You can download the database drivers from the following locations.

**Important**

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

<table>
<thead>
<tr>
<th>Database Engine</th>
<th>Drivers</th>
<th>Download Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Aurora (MySQL compatible)</td>
<td>mysql-connector-java-5.1.6.jar</td>
<td><a href="https://www.mysql.com/products/connector/">https://www.mysql.com/products/connector/</a></td>
</tr>
<tr>
<td>Amazon Aurora (PostgreSQL compatible)</td>
<td>postgresql-9.4-1204-jdbc42.jar</td>
<td><a href="https://jdbc.postgresql.org/download.html">https://jdbc.postgresql.org/download.html</a></td>
</tr>
<tr>
<td>Amazon Redshift</td>
<td>RedshiftJDBC41-1.1.10.1010.jar</td>
<td><a href="https://docs.aws.amazon.com/redshift/latest/mgmt/configure-jdbc-connection.html">https://docs.aws.amazon.com/redshift/latest/mgmt/configure-jdbc-connection.html</a></td>
</tr>
<tr>
<td>Greenplum Database</td>
<td>postgresql-9.4-1204-jdbc42.jar</td>
<td><a href="https://jdbc.postgresql.org/">https://jdbc.postgresql.org/</a></td>
</tr>
<tr>
<td>Maria DB</td>
<td>mariadb-java-client-2.4.0.jar</td>
<td><a href="https://mariadb.com/downloads/#connectors">https://mariadb.com/downloads/#connectors</a></td>
</tr>
<tr>
<td>MySQL</td>
<td>mysql-connector-java-5.1.6.jar</td>
<td><a href="https://www.mysql.com/products/connector/">https://www.mysql.com/products/connector/</a></td>
</tr>
</tbody>
</table>
## Installing JDBC Drivers on Linux

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

### To install JDBC drivers on your Linux system

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

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

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

### Database Engine | Installation Commands
---|---
Amazon Aurora (MySQL compatible) | `<PROMPT>` cd /usr/local/jdbc-drivers  
| | `<PROMPT>` sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz
Amazon Aurora (PostgreSQL compatible) | `<PROMPT>` cd /usr/local/jdbc-drivers  
| | `<PROMPT>` sudo cp -a /tmp/postgresql-X.X.X.jre7.tar
Microsoft SQL Server | `<PROMPT>` cd /usr/local/jdbc-drivers  
| | `<PROMPT>` sudo tar xzvf /tmp/sqljdbc_X.X.X_enu.tar.gz
MySQL | `<PROMPT>` cd /usr/local/jdbc-drivers  
| | `<PROMPT>` sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz
Oracle | `<PROMPT>` cd /usr/local/jdbc-drivers
### Storing Driver Paths in the Global Settings

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

#### To update the driver file locations

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

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

   **Note**
   For Teradata, you specify two drivers separated by a semicolon.
3. When you are finished adding the driver paths, choose **OK**.

**Updating the AWS SCT**

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

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

**To check for updates to AWS SCT**

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

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

Topics
- The AWS SCT Project Window (p. 12)
- Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13)
- Storing Database Passwords (p. 16)
- Using the Union All View for Projects with Partitioned Tables (p. 16)
- Using AWS SCT Tree Filters (p. 16)
- Hiding Schemas in the AWS SCT Tree View (p. 19)
- Keyboard Shortcuts for the AWS SCT (p. 19)
- Creating and Reviewing the Database Migration Assessment Report (p. 20)
- Starting the AWS Schema Conversion Tool (p. 24)
- Creating an AWS Schema Conversion Tool Project (p. 25)
- Converting Your Schema (p. 26)
- Applying the Converted Schema to Your Target DB Instance (p. 27)

The AWS SCT Project Window

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

1. In the left panel, the schema from your source database is presented in a tree view. Your database schema is "lazy loaded." In other words, when you select an item from the tree view, AWS SCT gets and displays the current schema from your source database.
2. In the top middle panel, action items appear for schema elements from the source database engine that couldn't be converted automatically to the target database engine.
3. In the right panel, the schema from your target DB instance is presented in a tree view. Your database schema is "lazy loaded." That is, at the point when you select an item from the tree view, AWS SCT gets and displays the current schema from your target database.
4. In the lower left panel, when you choose a schema element, properties describing the source schema element and the SQL command to create that element in the source database are displayed.

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

Using AWS Service Profiles in the AWS Schema Conversion Tool

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

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

**Storing AWS Credentials**

Use the following procedure to store AWS credentials globally.

**To store AWS credentials**

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

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

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

<table>
<thead>
<tr>
<th>AWS SCT Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema/Database Name</td>
<td>Choose the schema and database name that you want to filter on. For</td>
</tr>
<tr>
<td></td>
<td>schema name, you can enter “%” to select all schemas.</td>
</tr>
<tr>
<td>a. For Profile name,</td>
<td>type a name for your profile.</td>
</tr>
<tr>
<td>b. For AWS Access Key,</td>
<td>type your AWS access key.</td>
</tr>
<tr>
<td>c. For AWS Secret Key,</td>
<td>type your AWS secret key.</td>
</tr>
<tr>
<td>d. For Region,</td>
<td>choose the region for your profile.</td>
</tr>
</tbody>
</table>

Version 1.0
e. For **S3 Bucket**, choose the Amazon S3 bucket for your profile. You need to specify a bucket only if you are using a feature that connects to S3.

f. Choose **Use FIPS endpoint for S3** if you need to comply with the security requirements for the Federal Information Processing Standard (FIPS). FIPS endpoints are available in the following AWS Regions:

   • US East (N. Virginia) Region
   • US East (Ohio) Region
   • US West (N. California) Region
   • US West (Oregon) Region

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

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

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

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

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

To set the default profile for a project

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

Storing Database Passwords

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

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

Using 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. Choose a data warehouse (OLAP) project.
2. Choose **Settings**, and then choose **Project settings**. The **Current project settings** dialog box appears.
3. Choose **Use Union all view**.
4. Choose **OK** to save the settings and close the **Current project settings** dialog box.

Using AWS SCT Tree Filters

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

Some databases can have a large number of objects in the tree structure. You can use **tree filters** in AWS SCT to search for objects in the source and target tree structures. When you use a tree filter, you don't change the objects that are converted when you convert your database. The filter only changes what you see in the tree.
Tree filters work with objects that AWS SCT has preloaded. In other words, AWS SCT doesn't load objects from the database during searches. This approach means that the tree structure generally contains fewer objects than are present in the database.

For tree filters, keep the following in mind:

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

**To create a tree filter**

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

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

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

<table>
<thead>
<tr>
<th>Action</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choose the schema and database name you want to filter on. For schema name, you can enter “%” to choose all schemas.</strong></td>
<td><strong>Schema/Database Name</strong></td>
</tr>
<tr>
<td><strong>Choose the object name if you want to search by object name.</strong></td>
<td><strong>Object Name</strong></td>
</tr>
<tr>
<td><strong>Choose the table state. The options include the following:</strong></td>
<td><strong>Tree States</strong></td>
</tr>
<tr>
<td>- <strong>Any</strong> – Show all tables.</td>
<td></td>
</tr>
<tr>
<td>- <strong>Converted</strong> – Show only converted tables.</td>
<td></td>
</tr>
<tr>
<td>- <strong>Not Converted</strong> – Show only tables that have not been converted.</td>
<td></td>
</tr>
<tr>
<td>- <strong>Has Actions</strong> – Show tables with pending actions.</td>
<td></td>
</tr>
<tr>
<td><strong>Choose objects from the list of object types that you can filter on. Expand the tree nodes to load more objects to the object types list.</strong></td>
<td><strong>Object types list</strong></td>
</tr>
</tbody>
</table>

5. Choose **Apply**. After you choose **Apply**, the undo filter icon (next to the filter icon) is enabled. Use this icon if you want to remove the filters you applied.

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

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

### Importing a File List for the Tree Filter

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

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

The file to import should have the following format:

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

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

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

2. Choose the **Import File List** tab.
3. Choose **Import File**.
4. Choose a file to import, and then choose **Open**.
5. Choose **Apply**, and then choose **Close**.
Hiding Schemas in the AWS SCT Tree View

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

To hide databases and schemas in tree view

1. Open an AWS SCT project.
2. Connect to the data store that you want to show in tree view.
3. Choose **Settings, Global Settings, Tree View**.

![Tree View Settings](image)

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

Keyboard Shortcuts for the AWS SCT

The following are the keyboard shortcuts that you can use with the AWS SCT.
Creating and Reviewing the Database Migration Assessment Report

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

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

To create and view the database migration assessment report

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

2. Choose the **Action Items** tab.

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

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

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

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

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

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

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

Database Objects with Conversion Actions for MySQL

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

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

Figure: Conversion statistics for database storage objects

Detailed Recommendations for MySQL Migrations

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

Storage Object Actions

Constraint Changes

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

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

To create your project

1. Start the AWS Schema Conversion Tool.
2. Choose New Project from the File menu. The New Project dialog box appears.
3. Add the following preliminary project information.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>Type a name for your project, which is stored locally on your computer.</td>
</tr>
<tr>
<td>Location</td>
<td>Type the location for your local project file.</td>
</tr>
<tr>
<td>Source DB Engine (OLTP)</td>
<td>Choose Microsoft SQL Server, MySQL, Oracle, or PostgreSQL.</td>
</tr>
<tr>
<td>Target DB Engine (OLTP)</td>
<td>Choose Amazon Aurora (MySQL compatible), Amazon Aurora (PostgreSQL compatible), Amazon RDS for Microsoft SQL Server, Amazon RDS for MySQL, Amazon RDS for Oracle, or Amazon RDS for PostgreSQL.</td>
</tr>
<tr>
<td>Source DB Engine (OLAP)</td>
<td>Choose Amazon Redshift, Greenplum, Microsoft SQL Server DW, Netezza, Oracle DW, Teradata, or Vertica.</td>
</tr>
<tr>
<td>Target DB Engine (OLAP)</td>
<td>Choose Amazon Redshift.</td>
</tr>
</tbody>
</table>
4. Choose OK to create your AWS SCT project.
Converting Your Schema

Use the following procedure to convert schema.

**To convert schema**

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

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

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

At this point, no schema is applied to your target Amazon RDS DB instance. The planned schema is part of your project. If you select a converted schema item, you can see the planned schema command in the panel at lower center for your target Amazon RDS DB instance.
Applying the Converted Schema to Your Target DB Instance

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

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

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

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

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

1. Install AWS SCT. For more information, see Installing, Verifying, and Updating the AWS Schema Conversion Tool (p. 4).
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. 135).
3. Familiarize yourself with the user interface of AWS SCT. For more information, see Using the AWS Schema Conversion Tool User Interface (p. 12).
4. Create an AWS SCT project. Connect to your source and target databases. For more information about connecting to your source database, see Source Databases for the AWS Schema Conversion Tool (p. 30).
5. Run and then review the Database Migration Assessment Report. For more information about the assessment report, see Creating and Reviewing the Database Migration Assessment Report (p. 20).
6. Convert the source database schemas. There are several aspects of the conversion you need to keep in mind, such as what to do with items that don't convert, and how to map items that should be converted a particular way. For more information about converting a source schema, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 92).

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 the AWS Schema Conversion Tool (p. 106).

7. Applying the schema conversion to your target. For more information about applying a source schema conversion, see Using the AWS Schema Conversion Tool User Interface (p. 12).
8. The AWS SCT can also be used to convert SQL stored procedures and other application code. For more information, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 180).

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

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

Topics
- Using Oracle as a Source for AWS Schema Conversion Tool (p. 30)
- Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (p. 43)
- Using MySQL as a Source for AWS Schema Conversion Tool (p. 52)
- Using PostgreSQL as a Source for AWS Schema Conversion Tool (p. 54)
- Using Db2 LUW as a Source for AWS Schema Conversion Tool (p. 56)
- Using Amazon Redshift as a Source for AWS Schema Conversion Tool (p. 61)
- Using Oracle DW as a Source for AWS Schema Conversion Tool (p. 63)
- Using Teradata as a Source for AWS Schema Conversion Tool (p. 66)
- Using Netezza as a Source for AWS Schema Conversion Tool (p. 69)
- Using Greenplum as a Source for AWS Schema Conversion Tool (p. 71)
- Using Vertica as a Source for AWS Schema Conversion Tool (p. 72)
- Using Microsoft SQL Server DW as a Source for AWS Schema Conversion Tool (p. 74)

Using Oracle as a Source for AWS Schema Conversion Tool

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

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

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

For more information, see the following sections:

Topics
- Permissions Required When Using Oracle as a Source Database (p. 31)
- Connecting to Oracle as a Source Database (p. 31)
- Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL) (p. 33)
Permissions Required When Using Oracle as a Source Database

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

- CONNECT
- SELECT_CATALOG_ROLE
- SELECT ANY DICTIONARY
- SELECT on SYS.USER$ TO <sct_user>

Connecting to Oracle as a Source Database

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 **Connect to Oracle**.
   
   ![Connect to Oracle dialog box](image)

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

2. Provide the Oracle source database connection information. Use the instructions in the following table.
### Connecting to Oracle as a Source

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Choose the connection type to your database. Depending on your type, provide the following additional information:</td>
</tr>
<tr>
<td></td>
<td>- <strong>SID</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Server name</strong>: The DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Server port</strong>: The port used to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Oracle SID</strong>: The Oracle System ID (SID). To find the Oracle SID, submit the following query to your Oracle database:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>Service Name</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Server name</strong>: The DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Server port</strong>: The port used to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Service Name</strong>: The name of the Oracle service to connect to.</td>
</tr>
<tr>
<td></td>
<td>- <strong>TNS Alias</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>TNS file path</strong>: The path to the file that contains the Transparent Network Substrate (TNS) name connection information.</td>
</tr>
<tr>
<td></td>
<td>- <strong>TNS file path</strong>: The TNS alias from this file to use to connect to the source database.</td>
</tr>
<tr>
<td></td>
<td>- <strong>TNS Connect Identifier</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>TNS identifier</strong>: The identifier for the registered TNS connection information.</td>
</tr>
<tr>
<td><strong>User name</strong> and <strong>Password</strong></td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
</tbody>
</table>

The first time you connect to the Oracle database, you have to enter the path to the Oracle Driver file (ojdbc7.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). You must register on the free Oracle Technical Network website to complete the download. AWS SCT uses the selected driver for any future Oracle database connections. The driver path can be modified using the Drivers tab in Global Settings.

**Note**

AWS SCT uses the password to connect to your source database only when you create your project or choose the **Connect to source** option in a project, where **source** is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.
For This Parameter | Do This
---|---
**Use SSL** | Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:
  - **SSL Authentication**: Select this option to use SSL authentication by certificate is used for the connection instead of user name and password.
  - **Trust Store**: The location of a trust store containing certificates.
  - **Trust Store Password**: The password for the trust store.
  - **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.
  - **Trust Store Password**: The password for the key store. 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. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.

**Oracle Driver Path** | Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8).
If you store the driver path in the global project settings, the driver path doesn’t appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10).

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

**Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL)**

AWS SCT can convert SQL*Plus files into PSQL. The assessment report shows how AWS SCT converted the SQL*Plus files into PSQL. To convert SQL*Plus files to PSQL, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 180).

This section covers the following topics:

**Topics**
- Converting Oracle ROWID to PostgreSQL (p. 34)
- Converting Dynamic SQL for Oracle to PostgreSQL Migrations (p. 34)
- Converting Oracle Partitions to PostgreSQL Version 10 Partitions (p. 35)

When converting Oracle system objects to PostgreSQL, AWS SCT converts the following:
### Oracle system object | Description | Converted PostgreSQL object
---|---|---
V$VERSION | Displays version numbers of core library components in the Oracle Database | aws_oracle_ext.v $version
V$INSTANCE | A view that shows the state of the current instance. | aws_oracle_ext.v $instance

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

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

### Converting Oracle ROWID to PostgreSQL

In an Oracle database, the ROWID pseudocolumn contains the address of the table row. The ROWID pseudocolumn is unique to Oracle, so AWS SCT can convert the ROWID pseudocolumn to a data column when you convert to PostgreSQL. This conversion lets you retain the ROWID information.

When AWS SCT converts the ROWID pseudocolumn, it creates a data column with the data type BIGINT. 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.

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

1. Start AWS SCT. Choose a project with Oracle as the source.
2. Choose **Settings**, and then choose **Project settings**. The **Current project settings** dialog box appears.
3. Choose **true** for **Generate row ID**.
4. Choose **OK** to save the settings and close the **Current project settings** dialog box.

### Converting Dynamic SQL for Oracle to PostgreSQL Migrations

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

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

1. Create an Oracle-to-PostgreSQL migration project.
2. Connect to the source and target databases.
3. Choose a stored procedure in the Oracle source tree view. The procedure should contain references to the DBMS_SQL Oracle package or have an EXECUTE IMMEDIATE statement.
4. For **Actions**, choose **Convert Schema**, and agree to replace the objects if they exist. The following screenshot shows the converted procedure below the Oracle procedure.
Converting Oracle Partitions to PostgreSQL Version 10
Partitions

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

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

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

AWS SCT currently supports the following partitioning scenarios:

- Range
- List
- Range-Range
- List-List
The following scenarios are not currently supported:

- Hash
- Range-Hash
- List-Hash
- Interval
- Reference
- System

Some known issues with partition conversion to PostgreSQL version 10 include the following:

- Only not-null columns can be partitioned by columns.
- DEFAULT is not a possible value for a partition value.
- Partitions based on Timestamp with Timezone are not supported.
- Partitions based on a hash function are not supported.
- You can't update columns in partitioned tables because updating a value might cause the value to go to a different partition. AWS SCT only supports deleting and inserting columns in partitioned tables.
- Foreign keys are not supported going from and to partitioned tables.

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

The following are some things to consider regarding GOTO statements:

- A GOTO statement and a label can be used to change the order that statements are run in. Any PL/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.

Oracle GOTO statements and the converted MySQL statements

<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
</table>
| BEGIN  
....  
statement1;  
....  
GOTO label1;  
statement2;  
....  
label1:  
Statement3;  
....  | BEGIN  
label1:  
BEGIN  
....  
statement1;  
....  
LEAVE label1;  
statement2;  
....  
END;  
Statement3; |
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 may be more efficient as the data is easily retrieved from the temporary table rather than being required by each reference. This can be emulated by using additional views or a temporary table. The view name uses the format <procedure_name>$<subselect_alias>.

### Oracle WITH statements and the converted MySQL statements

<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE PROCEDURE TEST_ORA_PG.P_WITH_SELECT_VARIABLE_01 (p_state IN NUMBER) AS</td>
<td>CREATE PROCEDURE test_ora_pg.P_WITH_SELECT_VARIABLE_01(IN par_P_STATE DOUBLE)</td>
</tr>
<tr>
<td>BEGIN</td>
<td>BEGIN</td>
</tr>
<tr>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>

**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 may be more efficient as the data is easily retrieved from the temporary table rather than being required by each reference. This can be emulated by using additional views or a temporary table. The view name uses the format procedure_name$subselect_alias.

### Oracle WITH statements and the converted MySQL statements

<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE PROCEDURE TEST_ORA_PG.P_WITH_SELECT_VARIABLE_01 (p_state IN NUMBER) AS</td>
<td>CREATE PROCEDURE test_ora_pg.P_WITH_SELECT_VARIABLE_01(IN par_P_STATE DOUBLE)</td>
</tr>
<tr>
<td>BEGIN</td>
<td>BEGIN</td>
</tr>
<tr>
<td>END</td>
<td>END</td>
</tr>
</tbody>
</table>
### Oracle statement

```sql
l_dept_id NUMBER := 1;
BEGIN
FOR cur IN
  (WITH dept_empl(id, name, surname, lastname, state, dept_id)
  AS
    (SELECT id, name, surname, lastname, state, dept_id
    FROM test_ora_pg.dept_employees
    WHERE state = p_state AND dept_id = l_dept_id)
  SELECT id, state
  FROM dept_empl
  ORDER BY id)
  LOOP
    NULL;
END LOOP;
```

### MySQL statement

```sql
DECLARE var_l_dept_id DOUBLE DEFAULT 1;
DECLARE var$id VARCHAR (8000);
DECLARE var$state VARCHAR (8000);
DECLARE done INT DEFAULT FALSE;
DECLARE cur CURSOR FOR SELECT id, state
FROM (SELECT id, name, surname, lastname, state, dept_id
FROM test_ora_pg.dept_employees
WHERE state = par_p_state AND dept_id = var_l_dept_id) AS dept_empl
ORDER BY id;
DECLARE CONTINUE HANDLER FOR NOT FOUND
SET done := TRUE;
OPEN cur;
read_label:
  LOOP
    FETCH cur INTO var$id, var$state;
    IF done THEN
      LEAVE read_label;
    END IF;
    BEGIN
      END;
    END LOOP;
  CLOSE cur;
END;
```
<table>
<thead>
<tr>
<th>Oracle statement</th>
<th>MySQL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE PROCEDURE TEST ORA_PG.P_WITH_SELECT_REGULAR_MULT_01 AS</td>
<td>CREATE VIEW TEST ORA_PG.`P_WITH_SELECT_REGULAR_MULT_01$dept_empl (id, name, surname, lastname, state, dept_id) AS</td>
</tr>
<tr>
<td>BEGIN</td>
<td>(SELECT id, name, surname, lastname, state, dept_id FROM test_ora_pg.dept_employees WHERE state = 1);</td>
</tr>
<tr>
<td>FOR cur IN (</td>
<td>CREATE VIEW TEST ORA_PG.`P_WITH_SELECT_REGULAR_MULT_01$dept (deptid, parent_id, deptname) AS</td>
</tr>
<tr>
<td>WITH dept_empl AS</td>
<td>(SELECT id deptid, parent_id, name deptname FROM test_ora_pg.department);</td>
</tr>
<tr>
<td>(</td>
<td>CREATE PROCEDURE test_ora_pg.P_WITH_SELECT_REGULAR_MULT_01() BEGIN</td>
</tr>
<tr>
<td>SELECT id, name, surname, lastname, state, dept_id</td>
<td>DECLARE var$ID DOUBLE;</td>
</tr>
<tr>
<td>FROM test_ora_pg.dept_employees</td>
<td>DECLARE var$NAME VARCHAR (30);</td>
</tr>
<tr>
<td>WHERE state = 1),</td>
<td>DECLARE var$SURNAME VARCHAR (30);</td>
</tr>
<tr>
<td>dept AS</td>
<td>DECLARE var$LASTNAME VARCHAR (30);</td>
</tr>
<tr>
<td>(SELECT id deptid, parent_id, name deptname FROM test_ora_pg.department)</td>
<td>DECLARE var$STATE DOUBLE;</td>
</tr>
<tr>
<td>) LOOP</td>
<td>DECLARE var$DEPT_ID DOUBLE;</td>
</tr>
<tr>
<td>END LOOP;</td>
<td>DECLARE var$deptid DOUBLE;</td>
</tr>
<tr>
<td>END LOOP;</td>
<td>DECLARE var$PARENT_ID DOUBLE;</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>DECLARE var$deptname VARCHAR (200);</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>DECLARE done INT DEFAULT FALSE;</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>DECLARE cur CURSOR FOR SELECT dept_empl.<em>, dept.</em> FROM test_ora_pg.dept_employees AS dept_empl, test_ora_pg.dept AS dept</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>WHERE dept_empl.DEPT_ID = dept.DEPTID;</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>DECLARE CONTINUE HANDLER FOR NOT FOUND SET done := TRUE;</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>OPEN cur;</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>read_label: LOOP</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>FETCH cur INTO var$ID, var$NAME, var$SURNAME, var$LASTNAME, var$STATE, var$DEPT_ID, var$deptid, var$PARENT_ID, var$deptname;</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</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>
### 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 l_dept_id NUMBER := 10; BEGIN FOR cur IN ( WITH emp AS (SELECT id, name, surname, lastname, state, dept_id FROM test_ora_pg.dept_employees WHERE dept_id &gt; 10 ), active_emp AS (SELECT id FROM emp WHERE emp.state = p_state ) SELECT * FROM active_emp ) LOOP NULL; END LOOP; END;</td>
<td>CREATE VIEW TEST_ORA_PG.<code>P_WITH_SELECT_VAR_CROSS_01$emp</code>(id, name, surname, lastname, state, dept_id) AS (SELECT id, name, surname, lastname, state, dept_id FROM TEST_ORA_PG.DEPT_EMPLOYEES WHERE DEPT_ID &gt; 10); CREATE PROCEDURE test_ora_pg.P_WITH_SELECT_VAR_CROSS_02(IN par_P_STATE DOUBLE) BEGIN DECLARE var_l_dept_id DOUBLE DEFAULT 10; DECLARE var$ID DOUBLE; DECLARE done INT DEFAULT FALSE; DECLARE cur CURSOR FOR SELECT * FROM (SELECT ID FROM TEST_ORA_PG.<code>P_WITH_SELECT_VAR_CROSS_01$emp</code> AS emp WHERE emp.STATE = par_p_state) AS active_emp; DECLARE CONTINUE HANDLER FOR NOT FOUND SET done := TRUE; OPEN cur; read_label: LOOP FETCH cur INTO var$ID; IF done THEN LEAVE read_label; END IF; BEGIN END LOOP; CLOSE cur; END;</td>
</tr>
</tbody>
</table>
• 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 DPUMP utility, and so on.

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

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

You can grant all other privileges to an 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 you must 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 support 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. For more information, see Version 1.0.
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 `@dblink` to the table, view, or PL/SQL object name. You can query a table or view in the other database with the `SELECT` statement. For more information about using Oracle database links, see the Oracle documentation.

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

- 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 Unified Auditing, and 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 doesn’t support Oracle Unified Auditing. Amazon RDS for Oracle supports traditional auditing and fine-grained auditing using the DBMS_FGA package.
- 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 Microsoft SQL Server as a Source for AWS Schema Conversion Tool

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

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

For more information, see the following sections:

Topics
- Permissions Required When Using Microsoft SQL Server as a Source (p. 43)
- Using Windows Authentication When Using Microsoft SQL Server as a Source (p. 43)
- Connecting to SQL Server as a Source (p. 45)
- Converting SQL Server to MySQL (p. 46)
- Converting SQL Server to PostgreSQL (p. 48)
- Converting SQL Server to Amazon RDS for SQL Server (p. 51)

Permissions Required When Using Microsoft SQL Server as a Source

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

- VIEW DEFINITION
- VIEW DATABASE STATE

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

Using Windows Authentication When Using Microsoft SQL Server as a Source

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


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

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

### Using Windows Authentication with a JDBC Connection

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

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

To use integrated authentication

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

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

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

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

- need to declare two variables which point to the installed path of your JDBC:

  - variable name: SQLJDBC_HOME; variable value: `D:\lib\JDBC4.1\enu` (where your sqljdbc4.jar exists);

  - variable name: SQLJDBC_AUTH_HOME; variable value: `D:\lib\JDBC4.1\enu\auth\x86` (if you are running 32bit OS) or `D:\lib\JDBC4.1\enu\auth\x64` (if you are running 64bit OS). This is where your `sqljdbc_auth.dll` is located.
- copy `sqljdbc_auth.dll` to folder where your JDK/JRE is running. You may copy to lib folder, bin folder, etc. I copied to the following folder:

   ```
   [JDK_INSTALLED_PATH]\bin;
   [JDK_INSTALLED_PATH]\jre\bin;
   [JDK_INSTALLED_PATH]\jre\lib;
   [JDK_INSTALLED_PATH]\lib;
   ```
Connecting to SQL Server as a Source

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

Connecting to SQL Server as a Source

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

To connect to a Microsoft SQL Server source database

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

The Connect to Microsoft SQL Server dialog box appears.

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

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

Note

If you are running a 32-bit Java Virtual Machine (JVM), use the sqljdbc_auth.dll file in the x86 folder, even if the operating system is the x64 version. If you are running a 64-bit JVM on a x64 processor, use the sqljdbc_auth.dll file in the x64 folder.
<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server name</strong></td>
<td>Type the Domain Name Service (DNS) name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Instance name</strong></td>
<td>Type the instance name for the SQL Server database. To find the instance name, run the query <code>SELECT @@servername;</code> on your SQL Server database.</td>
</tr>
</tbody>
</table>
| **User name and Password** | Type the user name and password to connect to your source database server.  

**Note**  
AWS SCT uses the password to connect to your source database only when you create your project or choose the **Connect to source** option in a project, where **source** is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed. |
| **Use SSL** | Select this option if you want to use Secure Sockets Layer (SSL) to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:  
- **Trust Server Certificate**: Select this option to trust the server certificate.  
- **Trust Store**: The location of a trust store containing certificates.  
- **Trust Store Password**: The password for the trust store. |
| **Store Password** | AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password. |
| **Sql Server Driver Path** | Type the path to the driver to use to connect to the source database. For more information, see Installing the Required Database Drivers (p. 8).  
If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see Storing Driver Paths in the Global Settings (p. 10). |

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

**Converting 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.

<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 |
| BEGIN
  ....
  statement1;
  ....
  label1:
  statement2;
  ....
  GOTO label1;
  statement3;
  ....
  statement4;
  ....
  END | BEGIN
  ....
  statement1;
  ....
  label1:
  LOOP
  statement2;
  ....
  ITERATE label1;
  LEAVE label1;
  END LOOP;
  statement3;
  ....
  statement4;
  ....
  END |
| BEGIN
  ....
  statement1;
  ....
  label1:
  statement2;
  ....
  statement3;
  ....
  statement4;
  .... | BEGIN
  ....
  statement1;
  ....
  label1:
  BEGIN
  statement2;
  ....
  statement3;
  ....
  statement4; |
Converting SQL Server to PostgreSQL

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

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

  To ensure the uniqueness of index names, AWS SCT gives you the option to generate unique index names 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</td>
<td>BEGIN</td>
</tr>
<tr>
<td>..</td>
<td>label1:</td>
</tr>
<tr>
<td>statement1;</td>
<td>BEGIN</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>statement2;</td>
<td>statement1;</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>label1:</td>
<td>EXIT label1;</td>
</tr>
<tr>
<td>Statement3;</td>
<td>statement2;</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>END</td>
<td>END;</td>
</tr>
<tr>
<td></td>
<td>END</td>
</tr>
</tbody>
</table>

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

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

- **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, or zeros if the pattern is not found, on all valid text and character data types. When converting from SQL Server to Amazon RDS for PostgreSQL, AWS SCT replaces application code that uses `PATINDEX` with `aws_sqlserver_ext.patindex(<pattern character>, <expression character varying>)`.
- 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, or 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.SYSSYSTEMS</td>
<td>AWS_SQLSERVER_EXT.SYS_SYSTEMS</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>
<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.TABLE_CONSTRAINTS</td>
<td>AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLE_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>
</tbody>
</table>
Converting SQL Server Partitions to PostgreSQL Version 10 Partitions

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

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

Converting SQL Server to Amazon RDS for SQL Server

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

- AWS SCT can convert SQL Server Agent to provide schedules, alerts, and jobs on an Amazon RDS for SQL Server DB instance. After conversion, you can use an Amazon RDS for SQL Server DB instance as a data source for SQL Server Reporting Service (SSRS), SQL Server Analysis Services (SSAS), and SQL Server Integration Services (SSIS). You can’t run these services on the DB instance.
- 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 but you should review the behavior of objects that use link servers before you run the converted code.
- 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.
  - AlwaysOn is used. Amazon RDS does not support Always On.
  - Failover cluster is used.
• Database Mail is configured. Amazon RDS does not support Database Mail.
• SQL Server Reporting Service (SSRS) is used. Amazon RDS doesn't support SSRS so we recommend that you install SQL Server on an Amazon EC2 instance.
• SQL Server Analysis Services (SSAS) is used. Amazon RDS doesn't support SSAS so we recommend that you install SQL Server on an Amazon EC2 instance.
• SQL Server Integration Services (SSIS) is used. Amazon RDS doesn't support SSIS so we recommend that you install SQL Server on an Amazon EC2 instance.
• 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 MySQL as a Source for AWS Schema Conversion Tool

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

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

For more information, see the following sections:

Topics
• Privileges for MySQL as a Source Database (p. 52)
• Connecting to MySQL as a Source Database (p. 52)

Privileges for MySQL as a Source Database

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

• SELECT ON *.*
• SELECT ON mysql.proc
• SHOW VIEW ON *.*

Connecting to MySQL as a Source Database

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 Connect to MySQL.
The Connect to MySQL dialog box appears.

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

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

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

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

- **Require SSL**: Select this option if you want to connect to the server only through SSL.

  **Note**
  If you choose Require SSL, it means that if the server doesn't support SSL, you can't connect to the server. If you don't choose Require SSL and the server
Using PostgreSQL as a Source for AWS Schema Conversion Tool

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

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

For more information, see the following sections:

**Topics**

- Privileges for PostgreSQL as a Source Database (p. 54)
- Connecting to PostgreSQL as a Source (p. 55)

**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>`
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 **Connect to PostgreSQL**.

   ![Connect to PostgreSQL dialog box](image)

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

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

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

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

Use SSL | Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:
- **Verify Server Certificate**: Select this option to verify the server certificate by using a trust store.
- **Trust Store**: The location of a trust store containing certificates.
- **Trust Store Password**: The password for the trust store.

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

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

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

### Using Db2 LUW as a Source for AWS Schema Conversion Tool

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

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

### Permissions Needed When Using Db2 LUW as a Source

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

GRANT CONNECT ON DATABASE TO USER min_privs;

• Privilege needed to run SQL statements:

GRANT EXECUTE ON PACKAGE NULLID.SYSSH200 TO USER MIN_PRIVS;

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

• Privileges needed to check privileges granted through roles, groups and authorities:

  • GRANT EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID TO USER MIN_PRIVS;
  • GRANT EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID TO USER MIN_PRIVS;
  • GRANT EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSIBMADM.PRIVILEGES TO USER MIN_PRIVS;

• Privileges needed on system catalogs and tables:

  • GRANT SELECT ON SYSCAT.ATTRIBUTES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.CHECKS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.COLIDENTATTRIBUTES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.COLUMNS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.DATAPARTITIONEXPRESSION TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.DATAPARTITIONS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.DATATYPEDEP TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.DATATYPES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.HIERARCHIES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.INDEXCOLUSE TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.INDEXES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.INDEXPARTITIONS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.KEYCOLUSE TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.MODULEOBJECTS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.MODULES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.NICKNAMES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.PERIODS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.REFERENCES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.ROUTINEPARMS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.ROUTINES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.ROWFIELDS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.SCHEMATA TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.SEQUENCES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.TABCONST TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.TABLES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.TRIGGERS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.VARIABLEDEP TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.VARIABLES TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSCAT.VIEWS TO USER MIN_PRIVS;
  • GRANT SELECT ON SYSIBM.SYSDUMMY1 TO USER MIN_PRIVS;
• To run SQL statements, the user account needs a privilege to use at least one of the workloads enabled in the database. If none of the workloads are assigned to the user, ensure that the default user workload is accessible to the user:

GRANT USAGE ON WORKLOAD SYSDEFAULTUSERWORKLOAD TO USER MIN_PRIVS;

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

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

CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_8K
  PAGESIZE 8192
  BUFFERPOOL BP8K;

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

CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_BP16K
  PAGESIZE 16384
  BUFFERPOOL BP16K;

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

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

**Connecting to a Db2 LUW Source**

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

**To connect to a Db2 LUW source database**

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

   ![Connect to DB2 LUW](image)

   The **Connect to DB2 LUW** dialog box appears.
2. Provide the Db2 LUW source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td>Server port</td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td>Database</td>
<td>Type the name of the Db2 LUW database.</td>
</tr>
<tr>
<td>User name and Password</td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
</tbody>
</table>

**Note**

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

<table>
<thead>
<tr>
<th>Use SSL</th>
<th>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Verify Server Certificate</strong>: Select this option to verify the server certificate by using a trust store.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Trust Store</strong>: The location of a trust store containing certificates. You must add this location in Global Settings before it will appear here.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Trust Store Password</strong>: The password for the trust store.</td>
</tr>
<tr>
<td>For This Parameter</td>
<td>Do This</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>For more information about SSL support for Db2 LUW, see <a href="#">Configure Security Options for Connections</a>.</td>
</tr>
<tr>
<td>Store Password</td>
<td>AWS SCT creates a secure vault to store SSL certificates and database passwords. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.</td>
</tr>
<tr>
<td>DB2 LUW Driver Path</td>
<td>Type the path to the driver to use to connect to the source database. For more information, see <a href="#">Installing the Required Database Drivers</a> (p. 8). If you store the driver path in the global project settings, the driver path doesn't appear on the connection dialog box. For more information, see <a href="#">Storing Driver Paths in the Global Settings</a> (p. 10).</td>
</tr>
</tbody>
</table>

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

4. Choose **OK** to connect to your source database.

### Converting DB2 LUW to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL)

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

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

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

- Amazon Redshift

Privileges for Amazon Redshift as a Source Database

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

- USAGE ON SCHEMA `<schema_name>`
- SELECT ON ALL TABLES IN SCHEMA `<schema_name>`
- SELECT ON PG_CATALOG.PG_STATISTIC
- SELECT ON SVV_TABLE_INFO
- SELECT ON TABLE STV_BLOCKLIST
- SELECT ON TABLE STV_TBL_PERM

Connecting to Redshift as a Source

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

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

   ![Connect to Amazon Redshift dialog box](image)

   **The Connect to Amazon Redshift** dialog box appears.

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

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server name</strong></td>
<td>Type the DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Type the port used to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>Type the name of the Amazon Redshift database.</td>
</tr>
<tr>
<td><strong>User name and Password</strong></td>
<td>Type the user name and password to connect to your source database server.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>AWS SCT uses the password to connect to your source database only when you create your project or choose the <strong>Connect to source</strong> option in a project, where source is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.</td>
</tr>
<tr>
<td><strong>Use SSL</strong></td>
<td>Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the <strong>SSL</strong> tab:</td>
</tr>
</tbody>
</table>
Using Oracle DW as a Source for AWS Schema Conversion Tool

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

Privileges for Oracle Data Warehouse as a Source

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

- connect
- select_catalog_role
- select any dictionary

Connecting to Oracle DW 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 Connect to Oracle DW.
2. Provide the Oracle Data Warehouse source database connection information. Use the instructions in the following table.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Choose the connection type to your database. Depending on your type, provide the following additional information:</td>
</tr>
<tr>
<td></td>
<td>• <strong>SID</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>Server name</strong>: The DNS name or IP address of your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Server port</strong>: The port used to connect to your source database server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Oracle SID</strong>: The Oracle System ID (SID). To find the Oracle SID, submit the following query to your Oracle database:</td>
</tr>
<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>For This Parameter</td>
<td>Do This</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TNS file path</td>
<td>The path to the file that contains the Transparent Network Substrate (TNS) name connection information.</td>
</tr>
<tr>
<td>TNS file path</td>
<td>The TNS alias from this file to use to connect to the source database.</td>
</tr>
<tr>
<td>TNS Connect Identifier</td>
<td></td>
</tr>
<tr>
<td>TNS identifier</td>
<td>The identifier for the registered TNS connection information.</td>
</tr>
</tbody>
</table>

**User name and Password**

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

**Note**

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

**Use SSL**

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

- **SSL Authentication**: Select this option to use SSL authentication for the connection.
- **Trust Store**: The location of a trust store containing certificates.
- **Trust Store Password**: The password for the trust store.
- **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.
- **Trust Store Password**: The password for the key store. 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. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.

**Oracle Driver Path**

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

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

3. Choose Test Connection to verify that you can successfully connect to your source database.
4. Choose OK to connect to your source database.
Using Teradata as a Source for AWS Schema Conversion Tool

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

Privileges for Teradata as a Source

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

- SELECT ON DBC

Connecting to Teradata as a Source

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

To connect to a Teradata source database

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

   ![Connect to Teradata dialog box](image)

   The Connect to Teradata dialog box appears.

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

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

## Using LDAP Authentication with a Teradata Source

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

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

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

   ```xml
   AuthorizationSupported="no"
   LdapServerName="DC.test.local.com"
   ```
Applying the changes by running the configuration as follows.

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

2. 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
```

3. Restart TPA using the following command.

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

4. 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, you should specify this new password during connection to Teradata in LDAP mode. In DEFAULT mode, you still have to connect Teradata with the LDAP user name and any password.
Using Netezza as a Source for AWS Schema Conversion Tool

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

Privileges for Netezza as a Source

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

- SELECT ON SYSTEM.DEFINITION_SCHEMA.SYSTEM VIEW
- SELECT ON SYSTEM.DEFINITION_SCHEMA.SYSTEM TABLE
- SELECT ON SYSTEM.DEFINITION_SCHEMA.MANAGEMENT TABLE
- LIST ON <database_name>
- LIST ON <database_name>.ALL.TABLE
- LIST ON <database_name>.ALL.EXTERNAL TABLE
- LIST ON <database_name>.ALL.VIEW
- LIST ON <database_name>.ALL.MATERIALIZED VIEW
- LIST ON <database_name>.ALL.PROCEDURE
- LIST ON <database_name>.ALL.SEQUENCE
- LIST ON <database_name>.ALL.FUNCTION
- LIST ON <database_name>.ALL.AGGREGATE

Connecting to Netezza as a Source

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

To connect to a Netezza source database

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

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

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

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

4. Choose **OK** to connect to your source database.
Using Greenplum as a Source for AWS Schema Conversion Tool

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

Privileges for Greenplum as a Source

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

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

Connecting to Greenplum as a Source

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

To connect to a Greenplum source database

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

The Connect to Greenplum dialog box appears.

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

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

**Using Vertica as a Source for AWS Schema Conversion Tool**

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

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

- USAGE ON SCHEMA `<schema_name>`
- USAGE ON SCHEMA PUBLIC
- GRANT SELECT ON ALL TABLES IN SCHEMA `<schema_name>`
- SELECT ON ALL SEQUENCES IN SCHEMA `<schema_name>`
- EXECUTE ON ALL FUNCTIONS IN SCHEMA `<schema_name>`
- EXECUTE ON PROCEDURE `<schema_name.procedure_name(procedure_signature)>`

Connecting to Vertica as a Source

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

To connect to a Vertica source database

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

   ![Connect to Vertica dialog box](image)

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

2. Provide the Vertica source database connection information. Use the instructions in the following table.
Using Microsoft SQL Server DW as a Source for AWS Schema Conversion Tool

For This Parameter | Do This
--- | ---
Server name | Type the DNS name or IP address of your source database server.
Server port | Type the port used to connect to your source database server.
Database | Type the name of the Vertica database.
User name and Password | Type the user name and password to connect to your source database server.
  
  **Note**
  AWS SCT uses the password to connect to your source database only when you create your project or choose the Connect to source option in a project, where source is your source database. To guard against exposing the password for your source database, AWS SCT doesn't store the password. If you close your AWS SCT project and reopen it, you are prompted for the password to connect to your source database as needed.

Use SSL | Select this option if you want to use SSL to connect to your database. Provide the following additional information, as appropriate, on the SSL tab:
  
  • Trust Store: A trust store that you set up in the Global Settings.
  • Key Store: A key store that you set up in the Global Settings.

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.

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

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

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

Using Microsoft SQL Server DW as a Source for AWS Schema Conversion Tool

You can use AWS SCT to convert data from Microsoft SQL Server DW to Amazon Redshift.
Privileges for Microsoft SQL Server Data Warehouse as a Source

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

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

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

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

- VIEW SERVER STATE

Connecting to SQLServerDW as a Source

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

To connect to a Microsoft SQL Server data warehouse source database

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

   ![Connect to Microsoft SQL Server DW](image)

   The Connect to Microsoft SQL Server DW dialog box appears.

2. Provide the Microsoft SQL Server data warehouse source database connection information. Use the instructions in the following table.
### For This Parameter | Do This
--- | ---
**Server name** | Type the Domain Name Service (DNS) name or IP address of your source database server.

**Server port** | Type the port used to connect to your source database server.

**Instance name** | Type the instance name for the SQL Server database. To find the instance name, run the query `SELECT @@servername;` on your SQL Server database.

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

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

- **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. Enabling this option lets you store the database password and to connect quickly to the database without having to enter the password.

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

3. Choose **Test Connection** to verify that you can successfully connect to your source database.
4. Choose **OK** to connect to your source database.
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 or AWS Workload Qualification Framework (AWS WQF).

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 Schema Conversion Tool, and what you need else you need to do to complete the conversion. To create an assessment report, use Create Report from the context (right-click) menu of the database in AWS SCT.

You can use AWS WQF during the planning phase of your migration to learn what level of effort is required to migrate your data and applications. WQF is a standalone application that is integrated with AWS SCT and AWS DMS. You use it to create inventory reports and cost calculator reports. To set up WQF, go to Global Settings in the SCT application and set up the required JDBC drivers for your source databases. Then, start WQF and follow the instructions on the screen that opens in AWS Cloud9.

Creating Assessment Reports with AWS Schema Conversion Tool

An important part of the AWS Schema Conversion Tool is the database migration assessment report that it generates to help you convert your schema. The report summarizes all of the schema conversion tasks and details the action items for schema that can't be converted to the DB engine of your target DB instance. You can view the report in the application. To do so, export it as a comma-separated value (CSV) or PDF file.

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.
- Current source hardware configuration
- Recommendations, including conversion of server objects, backup suggestions, and linked server changes

The report includes information about an Amazon RDS DB instance if you selected Amazon RDS as your target, including the following:

- The currently used storage size and maximum storage size for the DB instance.
- The current number of databases on the DB instance and the maximum number of databases allowed on the DB instance.
- A list of database services and server objects that are not available on the DB instance.
- A list of databases that are currently participating in replication. Amazon RDS doesn't support replication.
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. This Estimated Complexity field is exported in the PDF version of the assessment report, but it's not included in the CSV version.

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.

You can find more details in the following topics:

Topics
- Creating a Database Migration Assessment Report (p. 78)
- Viewing the Assessment Report (p. 79)
- Saving the Assessment Report (p. 80)

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.
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. 79)
- Assessment Report Action Items (p. 80)

**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.

For schema items that can't be converted automatically to the target database engine, the summary includes an estimate of the effort required to create schema items in your target DB instance that are equivalent to those in your source.

The report categorizes the estimated time to convert these schema items as follows:
• **Simple** – Actions that can be completed in less than one hour.
• **Medium** – Actions that are more complex and can be completed in one to four hours.
• **Significant** – Actions that are very complex and take more than four hours to complete.

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

**Assessment Report Action Items**

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

The report also contains recommendations for how to manually convert the schema item. For more information about deciding how to handle manual conversions, see **Handling Manual Conversions in the AWS Schema Conversion Tool** (p. 99).

![Image of AWS Schema Conversion Tool interface](image)

**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.
Creating Migration Reports with the Workload Qualification Framework

AWS Workload Qualification Framework (AWS WQF) is a standalone app that is included with AWS SCT. You can use WQF to analyze your migration to the AWS Cloud. It assesses and rates the workload for the entire migration, including database and app modifications. WQF can recommend strategies and tools that you can use for your migration, and give you feedback that you can use to make changes. It can also identify actions that you need to take on a database to complete a migration to Amazon RDS or Amazon Aurora.

You can use WQF for the following migration scenarios:

- Oracle to Amazon RDS for PostgreSQL or Aurora with PostgreSQL compatibility
- Oracle to Amazon RDS for MySQL or Aurora with MySQL compatibility
- Microsoft SQL Server to Amazon RDS PostgreSQL or Aurora PostgreSQL

You can use WQF during the planning phase of your migration process to determine what you need to do to migrate your data and apps. SCT accesses your schema conversion; in contrast, WQF reports on the following:

- Workload assessment based on complexity, size, and technology used
- Recommendations on migration strategies
- Recommendations on migration tools
- Feedback on what exactly to do
- Assessment of the effort required based on the number of people on the migration project

Topics
- Setting Up AWS WQF (p. 82)
- Preparing for Batch Imports of Logical and Physical Components (p. 83)
- Creating WQF Reports (p. 85)
- Understanding Workload Categories (p. 89)

Setting Up AWS WQF

AWS WQF comes in an Amazon Machine Image (AMI) that is preconfigured with Microsoft Windows Server, AWS SCT, and WQF preinstalled. You can connect to it by using a remote desktop app.

Use the following procedure to launch the shared AMI. Before you begin, make sure that you have an active key pair stored in a .pem file on your local drive.

To launch the shared AMI and set up WQF

1. Choose one of the following:
   - Visit the AWS Marketplace Workload Qualification Framework page and launch the AMI in your account.
   - Search for Workload Qualification Framework in the EC2 Launch Wizard to find the AMI. Then launch it in your account.

2. Choose an instance of type m5a.4xlarge or larger, and add at least 100 GiB to its storage. You need your .pem file for this step.

   If you're not familiar with launching AMIs or creating EC2 instances, watch this brief AWS support video: How do I launch an EC2 instance from a custom Amazon Machine Image (AMI)

3. In the EC2 console, choose Instances. When the instance is available, open its context (right-click) menu, and choose Get Windows Password. Retrieve the password using the same .pem file that you used in the previous step to set up the instance.

4. Connect to the instance using a remote desktop app. For more information, see Connect to Your Windows Instance.

After SCT and WQF are set up, you need to set up your database drivers in SCT before you're ready to use WQF. Use the following procedure to set up the drivers.

1. Locate and download the Java Database Connectivity (JDBC) drivers for your data sources. Put the .jar files on the machine that has SCT and WQF installed. For more information, see Installing the Required Database Drivers (p. 8).
Preparing for Batch Imports of Logical and Physical Components

If you are analyzing a lot of servers, you can simplify the process by doing a batch import of both logical and physical components. To do this, you can gather information before using WQF. This way you don't have to manually enter the information for each server into the UI. This works well in situations where WQF doesn't have access to the servers. You can also use this method in online mode, where WQF can access the servers.

To run in batch mode, you need to generate two files:

- A JSON file containing logical components
- A comma-separated value (CSV) file containing physical components

You can use any method that you prefer to create these files. For both files, if you run WQF entirely offline, you don't need to supply the user name or password. This is because in offline mode you don't connect to any servers. In the following sections, we supply templates for these files.

**Note**
There are two separate workflows for creating components. Either upload an AWS SCT project .zip file, or upload a fleet-wide analysis by using JSON and CSV files. You don't need to do both.

Sample Template for Logical Components for Batch Import (JSON)

The following examples show JSON that you can use for the logical components for WQF batch import in offline mode and online mode.

The following example shows JSON that you can use when running WQF in offline mode. In offline mode, you can remove username, password, and connection-type from your JSON. You also add one additional field to your JSON. The extra field is named sct-file, and the values in it are the complete directory path and file name of the statistics .zip file generated by SCT. An example is the following: D:\Temp\WQF\Oracle-WqfStats-2019-03-09-01-56.zip. For offline mode, the JSON example looks like the following.

```json
{
    "name": "component_name",
    "description": "My description",
    "type": "database",
    "properties": {
        "number-of-advance-config-features": "123",
        "has-proprietary-logic-in-OLAP": "true",
        "number-of-hardware-features": "100"
    },
    "selected-schemas": [
        {
            "schema": "schema_name"
        }
    ]
}
```
The following example shows JSON that you can use for logical components when running WQF in online mode.

```json
[
  {
    "name": "component_name",
    "description": "My description",
    "type": "database",
    "properties": {
      "number-of-advance-config-features": "123",
      "has-proprietary-logic-in-OLAP": "true",
      "number-of-hardware-features": "100"
    },
    "selected-schemas": [
      {
        "schema": "schema_name"
      }
    ],
    "source-dialect": "ORACLE",
    "source-connection-config": {
      "server": "host",
      "server-port": "port",
      "sid": "ORCL"
    }
  },
  {
    "name": "component_name1",
    "description": "My new Description",
    "type": "database",
    "properties": {
      "number-of-advance-config-features": "321",
      "has-proprietary-logic-in-OLAP": "true",
      "number-of-hardware-features": "50"
    },
    "selected-schemas": [
      {
        "database": "db_name",
        "schema": "schema_name"
      }
    ],
    "source-dialect": "MSSQL",
    "source-connection-config": {
      "server": "host",
      "server-port": "port",
      "sid": "",
      "username": "user",
      "password": "pass"
    }
  },
  {
    "name": "component_name2",
    "description": "description",
    "source-dialect": "ORACLE",
    "source-connection-config": {
      "server": "host",
      "server-port": "port",
      "sid": "ORCL"
    },
    "sct-file": "D:\Temp\WQF\Oracle-WqfStats-2019-03-09-01-56.zip"
  }
]
```
Sample Template for Physical Components for Batch Import (CSV)

The following examples show a .csv file you can use for the physical components.

If you open this .csv file in Microsoft Excel, you can see the fields laid out. Doing this might make it easier to edit if you are not exporting your physical components programmatically. As a spreadsheet, the file looks similar to the following.

**Note**

If you are running in offline mode, you need to add one additional field to the following CSV. The extra field should be named `SCTFILE`, and the values in it should be the complete directory path and file name of the statistics zip file generated by SCT. An example is `D:\Temp\WQF\Oracle-WqfStats-2019-03-09-01-56.zip`.

The .csv file is as follows.

| NAME, serverName, port, instanceName/SID, userName, password, databaseType, description, hasHADRRequirements, hasHeavyGeospatialPayload | component_name, host, port, sid, user, password, Oracle, mycomponent, true, false |
| component_name, port, sid, user, password, Oracle, mycomponent, true, false |

The same file, shown in spreadsheet format, is as follows.

<table>
<thead>
<tr>
<th>NAME</th>
<th>serverName</th>
<th>port</th>
<th>instanceName</th>
<th>SID</th>
<th>userName</th>
<th>password</th>
<th>databaseType</th>
<th>description</th>
<th>hasHADRRequirements</th>
<th>hasHeavyGeospatialPayload</th>
</tr>
</thead>
<tbody>
<tr>
<td>component_name</td>
<td>host</td>
<td>port</td>
<td>sid</td>
<td>user</td>
<td>password</td>
<td>Oracle</td>
<td>mycomponent</td>
<td>TRUE</td>
<td>false</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

Creating WQF Reports

To create migration reports with WQF, take the following steps:
1. Create a WQF project.
2. Add an app or apps to the project.
3. Add a physical component or components for each app in the project. Doing this allows WQF to collect details about your source database server.

   You can use either a local installation of SCT or a .zip file from another machine.
4. Add a logical component or components for each app in the project. Doing this allows WQF to collect details about database objects and apps.

   You can use either a local installation of SCT or a .zip file from another machine.
5. Generate the inventory and WQF analysis reports to plan the project.
6. Copy the downloaded report files to a machine on which Microsoft Excel is installed and open them to get details about the migration project.

You can choose the following types of reports:

- **Inventory reports**

  These reports list feature-based and hardware information about the database server you're trying to migrate from. The data required for this report is collected by using physical components, where users can input details of a database server to collect information from. You can use information from this report to help plan the migration of a database server.

- **Cost calculator reports**

  These reports categorize the workload you're trying to migrate. They detail items for all the logical and physical components that were selected as part of the app. The report includes an alterable migration cost model based on the findings from SCT and recommendations on how to perform migrations per app along with steps to migrate. The cost calculator report also includes an inventory report.

You can view these reports in Microsoft Excel, or any app that supports the Office Open XML format.

**Topics**

- Starting WQF (p. 86)
- Creating and Editing a WQF Project (p. 86)

**Starting WQF**

After you set up SCT and WQF, start the WQF service by double-clicking on the WQF Start icon in the desktop. A series of terminal windows open as WQF starts.

After WQF has successfully started, use Google Chrome (preinstalled on the AMI) and navigate to http://localhost:4040/ide.html. Doing this opens the WQF module in the AWS Cloud9 console.

**Creating and Editing a WQF Project**

The WQF project provides a container function for application components, like databases, physical machines and the application code. You can save and come back to the WQF project later. You can maintain multiple migration projects at a time.

**To create or view a migration project and reports**

1. To create a new project, open WQF and enter a descriptive name for your project in Create new project. Then choose Create.
You can also choose an existing project at left.

2. A new screen displays. Choose one of the following:

- To start the report wizard, choose **Create report**. This wizard guides you through all required steps to use the WQF application.
- To access an existing report, choose it from the list. You can view, edit, or delete a report, or change its report type.

3. On the **Report** screen, choose a report type for **Type**. Enter a name and optional description for your report.

Choose **Next** to continue.

4. On the **Physical Components** tab, provide data about the physical components of your database system or fleet.

Physical component analysis helps collect details about the source database server for the inventory report. Based on advanced feature usage, WQF includes some details to calculate overall categorization of the migration project. You can choose to create a single physical component by using SCT (or by uploading a .zip file of an SCT project). Or you can upload details of multiple physical components as a JSON file to perform fleet-wide physical component analysis.

Choose one of the following:

- To connect to new physical components, choose **New**. Enter a descriptive name and optional description, and choose a source database. If you want to provide more information, follow the screen prompts. Choose **Next**.
- To import a CSV file, choose **Import**. For more information about the file format, see **Sample Template for Physical Components for Batch Import (CSV)** (p. 85).

On the **Import physical components from CSV file** screen, choose **Choose** to navigate to your CSV file, then choose **Save**. Choose **Cancel** to return to the **Physical Components** screen without making changes.

Choose **Next** to continue.

5. On the **Data Collection** panel of the **Physical Components** tab, choose whether to load your data structures from an archive file or from the database.

Choose one of the following:

- To import a ZIP file, choose **Load from Archive**. Choose **Choose** to navigate to your CSV file, then choose **Open**. For more information about the file format, see **Sample Template for Physical Components for Batch Import (CSV)** (p. 85).
- To connect to your database (default), choose **Load from DB** and enter the connection details:
  - Server name or IP address
  - Server port
  - Instance name (optional)
  - User name
  - Password
  - Whether to enable or not enable Secure Sockets Layer (SSL)

To perform data collection now, choose **Get schemas** and choose the schemas to include, then choose **Save**.

Repeat this step until you have loaded all your physical components. Then choose **Next** to continue.
6. (Only for Cost Calculator reports) On the **Logical Components** tab, you can load the logical components of your system. Logical component analysis helps WQF collect information about the following:

- **Database objects**: Analyze and categorize complexity of converting database objects to the target of choice
- **Application code**: Analyze and categorize embedded SQL in application code for conversion to the target of choice

You can choose to create a single logical component by using SCT (or by uploading a zip file of an SCT project). Alternatively, you can upload details of multiple logical components as a JSON file to perform fleet-wide logical component analysis.

Choose one of the following:

- To connect to new physical components, choose **New**. Enter a descriptive name and optional description, and choose a type and source database. If you want to provide more information, follow the screen prompts. Choose **Next**.
- To import a CSV file, choose **Import**. For more information about the file format, see *Sample Template for Logical Components for Batch Import (JSON)* (p. 83).

On the **Import physical components from CSV file** screen, choose **Choose** to navigate to your CSV file, then choose **Save**. If you want to close this screen without making changes, choose **Cancel** to return to the Physical Components screen.

7. (Only for Cost Calculator reports) On the **Data Collection** panel of the **Logical Components** tab, you choose whether to load your data structures from an archive file or from the database.

Choose one of the following:

- To import a ZIP file, choose **Load from Archive**. Choose **Choose** to navigate to your CSV file, then choose **Open**.
- To connect to your database (default), choose **Load from DB** and enter the connection details:
  - **Server Name or IP**
  - **Server Port**
  - **Instance Name** (optional)
  - **User Name**
  - **Password**
  - **SSL** (enable or not enable)

Choose **Run** to perform data collection now, or choose **Save** to save changes without running the process. If you are running in offline mode, you don’t need to run the data collection process, because everything is in the statistics file generated by SCT.

Repeat this step until you have loaded all your logical components. Then choose **Next** to continue.

8. (Only for Cost Calculator reports) On the **Applications** tab, you can select the components of your app. Enter a name for the application. On the left, choose the components you want to include. To remove components, choose them on the right to move them out of the **Selected Components** list.

Applications consist of a combination of physical and logical components. By analyzing these together, WQF determines the migration complexity based on different parameters and details collected as part of individual physical and logical component analysis.

Choose **Save** to continue.

9. On the **Report Configuration** tab, choose the components you want to include in your report.
Choose Next to continue.

10. On the Report Configuration tab, save your report by choosing Save. If you want to change something before saving, choose Previous.

11. On the All Reports tab, you can view all your report. To download a report, choose Download or Save.

   The report downloads in Open XML format, and you can open it in an Open XML viewer like Microsoft Excel. In the report, you can see the formatted categorization, analysis, and migration strategies. Reports are generated for each application, and contain chosen set of logical and physical components.

12. (Optional) To delete a report or a project, choose the report and then choose Delete.

Understanding Workload Categories

AWS WQF evaluates your migration workload, and classifies it into a workload category that characterized by the way your database and app are architected. Based on this categorization, WQF analyzes the components your system uses and extrapolates the type of work needed to do the migration. Based on this analysis, AWS WQF estimates how easy or difficult you can expect the migration to be. It also estimates the type of work involved and the level of effort required.

Topics
- Category 1: Workloads That Use ODBC and JDBC (p. 89)
- Category 2: Workloads with Light Use of Proprietary Features (p. 89)
- Category 3: Workloads with Heavy Use of Proprietary Features (p. 90)
- Category 4: Engine-Specific Workloads (p. 90)
- Category 5: Nonportable, Unacceptable Risk, or "Lift and Shift" Workloads (p. 90)

Category 1: Workloads That Use ODBC and JDBC

This category typically has fewer than 50 custom stored procedures, or has simple stored procedures that are used for access controls. Applications using this data connect to the database using Open Database Connectivity (ODBC) or Java Database Connectivity (JDBC) instead of using proprietary drivers that have nonstandard extensions. Application logic resides in code outside the database (Java, Python, Ruby, and so on). For these databases, there is either no requirement for supporting a read-replica or Multi-AZ deployment, or these features are offered through replication-based technologies.

In this category, data warehouses use a star or snowflake schema with a reporting layer that uses engine-specific SQL or ANSI SQL, like Amazon QuickSight or Tableau. Porting to Amazon Redshift is relatively straightforward because the data model is retained, and enhanced with defining sort keys, distribution keys, compression, and properly configuring workload management (WLM).

These workloads are easy to port to Amazon Aurora and Amazon RDS. A migration in this category usually requires few person-hours.

Category 2: Workloads with Light Use of Proprietary Features

Workloads in this category use a combination of app code (Java, Python, Ruby, and so on) and stored procedure code. Stored procedures are used when the logic is cumbersome to implement in app code. Generally, this type of workload has less than 200 stored procedures and doesn't use advanced SQL language features. Schema migration is simple because data structures such as tables and views are used.
In this category, data warehouse workloads can stage data in tables and transform it using SQL wrapped in simple stored procedures. Data warehouse writes might have some microbatching or a large number of updates, deletes, and transactions. The data warehouse can also use proprietary online analytical processing (OLAP) extensions such as CUBE, ROLLUP, or PIVOT.

Migration involves moving the stored procedure logic outside the database and reworking SQL reports to deal with the lack of native functions. These are relatively easy to migrate. You can expect the migration of this type of workload to consume a moderate number of person-hours.

Category 3: Workloads with Heavy Use of Proprietary Features

Workloads in this category are completely driven by advanced stored procedure logic or proprietary features. In the field, many of the workloads in this category have as many as 100,000 lines of database-resident code and features. These workloads also use advanced features such as virtual private databases, column obfuscation, tuning options, and user-defined types. They consume a large amount of time for translation into alternate execution environments. Some of these workloads rely on native hardware features, such as Exadata, Supercluster, and PDW. High-performance workloads often fall into this category. The tuning options present in local code have to be translated and tested with options available on the target database.

Data warehouses in this category contain large numbers of stored procedures and user-defined functions that either orchestrate extract, transform, load (ETL) operations or create business views. Their ETL processes can't be easily expressed in Amazon Redshift, although much of the business logic might be rendered as views. These data warehouses can also have many thousands of tables with a large number of transactions to manage the ETL workflow. When migrating with Amazon Redshift as a target, such workloads require rearchitecting the app to separate the transactional workload from reporting. Rearchitecting also requires pulling logic out of the data warehouse and into another compute layer.

These workloads are difficult to migrate and might constitute a material risk to the customer. You can expect the migration of this workload to consume a significant number of person-hours.

Category 4: Engine-Specific Workloads

Workloads in this category use frameworks that can work only with a specific commercial database engine. For example, database-specific app frameworks include Oracle Forms, Oracle Reports, Oracle ADF and Oracle APEX (Application Express), or apps that use .NET ActiveRecord extensively. Migrating these workloads to an open-source or NoSQL database can require a complete reimplementation of the app to separate presentation logic from the database.

A data warehouse in this category might rely heavily on proprietary features such as Geospatial at petabyte scale. These features might contain proprietary logic in OLAP data structures. Workloads might have availability, replication, or user concurrency requirements that can't be met by a single Availability Zone architecture. They might have latency requirements that preclude the use of Amazon Athena.

These workloads are very difficult to migrate. You can expect the migration of this workload to take a very large number of person-hours. Such a migration can also constitute a significant risk to undertake. The migration of this workload might not be supported from the perspective of certification or third-party support.

Category 5: Nonportable, Unacceptable Risk, or "Lift and Shift" Workloads

Workloads in this category might be implemented on database engines that have no cloud-based equivalent. Their underlying operating system might not be supported by AWS. For example, it might use mainframe, Power, or RISC architectures. In some cases, the database might use native code extensions such as Oracle Call Interface to run the business logic. This business logic is often considered "legacy" by the customer, even if it is still business-critical. In some cases, customers don't have the
source code for these programs. Data warehouse and OLTP workloads share the same attributes for this category.

You can migrate these apps to Amazon EC2. They might have emulation requirements or require other third-party solutions. In some cases, the risk of moving these workloads from the existing environment might be too high to justify. In that case, it's appropriate to maintain high-performance connectivity to the on-premises implementation with a network topology that supports the app requirements.
You can use the AWS Schema Conversion Tool (AWS SCT) to convert your existing database schemas from one database engine to another. Converting a database using the AWS SCT user interface can be fairly simple, but there are several things you need to consider before you do the conversion.

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

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

This section includes the following topics:

**Topics**
- Creating Mapping Rules in the AWS Schema Conversion Tool (AWS SCT) (p. 93)
- Converting Your Schema by Using the AWS Schema Conversion Tool (p. 95)
- Handling Manual Conversions in the AWS Schema Conversion Tool (p. 99)
- Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool (p. 100)
- Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 101)
- Comparing Database Schemas (p. 104)
- Finding Related Transformed Objects (p. 105)

AWS SCT supports the following OLTP conversions.

<table>
<thead>
<tr>
<th>Source Database</th>
<th>Target Database on Amazon RDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server (version 2008 and later)</td>
<td>Amazon Aurora (MySQL or PostgreSQL), Microsoft SQL Server, MySQL, PostgreSQL</td>
</tr>
<tr>
<td>MySQL (version 5.5 and later)</td>
<td>Amazon Aurora (PostgreSQL), MySQL, PostgreSQL</td>
</tr>
<tr>
<td>Oracle (version 10.2 and later)</td>
<td>Amazon Aurora (MySQL or PostgreSQL), MySQL, Oracle, PostgreSQL</td>
</tr>
</tbody>
</table>

You can migrate schema and data from MySQL to an Amazon Aurora (MySQL) DB cluster without using AWS SCT. For more information, see Migrating Data to an Amazon Aurora DB Cluster.
Creating Mapping Rules

Source Database | Target Database on Amazon RDS
---|---
PostgreSQL (version 9.1 and later) | Amazon Aurora (MySQL), MySQL, PostgreSQL
DB2 LUW (versions 9.5, 9.7, 10.5, and 11.1) | Amazon Aurora (MySQL or PostgreSQL), MySQL, PostgreSQL

If you want to convert a data warehouse schema, see Converting Data Warehouse Schemas to Amazon Redshift Using the AWS Schema Conversion Tool (p. 106).

To convert your database schema to Amazon RDS, you take the following high-level steps. Each step is a link to a section that provides more information.

- **Creating Mapping Rules in the AWS Schema Conversion Tool (p. 109)** – 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 the AWS Schema Conversion Tool (p. 95)** – 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 Assessment Reports with AWS Schema Conversion Tool (p. 77)** – 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 the AWS Schema Conversion Tool (p. 99)** – 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 the AWS Schema Conversion Tool (p. 100)** – You can update your AWS SCT project with the most recent schema from your source database.
- **Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 101)** – When you are ready, have AWS SCT apply the converted schema in your local project to your target Amazon RDS DB instance.

Creating Mapping Rules in the AWS Schema Conversion Tool (AWS SCT)

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

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

You can create mapping rules that perform the following tasks:

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

- Database
- Schema
- Table
- Column

Creating Mapping Rules

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

To create mapping rules

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

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

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

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

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

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

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

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

Viewing Mapping Rules for Objects

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

Exporting Mapping Rules

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

To export mapping rules

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

Converting Your Schema by Using the AWS Schema Conversion Tool

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

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

Converting Schema

To convert schema from your source database, choose a schema object to convert from the left panel of your project. Open the context (right-click) menu for the object, and then choose Convert schema, as shown following.
After you have converted the schema from your source database, you can choose schema items from the left panel of your project and view the converted schema in the center panels of your project. The lower-center panel displays the properties of and the SQL command to create the converted schema, as shown following.
Editing Converted Schema

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 Assessment Reports with AWS Schema Conversion Tool (p. 77).

When AWS SCT generates a converted schema, it doesn't immediately apply it to the target DB instance. Instead, the converted schema is stored locally until you are ready to apply it to the target DB instance. For more information, see Applying Your Converted Schema (p. 102).

Editing Converted Schema

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

To edit converted schema

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

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

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

**Clearing a Converted Schema**

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

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

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

- Modify your source database schema.
- Modify your target database schema.

Modifying Your Source Schema

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

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

Modifying Your Target Schema

For some items, it might be easier to apply the converted schema to your target database, and then add equivalent schema items manually to your target database for the items that couldn’t be converted automatically. You can write all of the schema that can be converted automatically to your target DB instance by applying the schema. For more information, see Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 101).

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 the AWS Schema Conversion Tool

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

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

When the AWS Schema Conversion Tool generates converted schema (as shown in Converting Your Schema by Using the AWS Schema Conversion Tool (p. 95)), 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 Assessment Reports with AWS Schema Conversion Tool (p. 77).

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

Saving Your Converted Schema to a File

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

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

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

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

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

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

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

For more information, see Using the AWS Lambda Functions from the AWS SCT Extension Pack (p. 188).
Comparing Database Schemas

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

The following schema comparisons are supported:

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

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

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

To compare two schemas

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

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

Finding Related Transformed Objects

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

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

To view related objects that were created during a schema conversion

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

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.

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

If you want to convert an online transaction processing (OLTP) database schema, see Converting Database Schemas Using the AWS Schema Conversion Tool (p. 92).

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

1. Specify the optimization strategy and rules, and specify the mapping that you want AWS SCT to use. You can set up rules that change the data type of columns, move objects from one schema to another, and change the names of objects.
   
   You can specify optimization and mapping in Settings. For more information on optimization strategies, see Choosing Optimization Strategies and Rules for Use with the AWS Schema Conversion Tool (p. 107). For more information about mapping, see Creating Mapping Rules in the AWS Schema Conversion Tool (p. 109).

2. Provide statistics on your source data warehouse so that AWS SCT can optimize how your data warehouse is converted. You can either collect statistics directly from the database, or upload an existing statistics file. For more information about providing data warehouse statistics, see Collecting or Uploading Statistics for the AWS Schema Conversion Tool (p. 108).

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 Assessment Reports with AWS Schema Conversion Tool (p. 77).

4. Convert the schema. AWS SCT creates a local version of the converted schema for you to review, but it doesn’t apply it to your target database until you are ready. For more information about converting, see Converting Your Schema by Using the AWS Schema Conversion Tool (p. 111).

5. After you convert your schema, you can manage and edit your keys. Key management is the heart of a data warehouse conversion. For more information about managing keys, see Managing and Customizing Keys in the AWS Schema Conversion Tool (p. 115).
Choosing Optimization Strategies and Rules for Use with the AWS Schema Conversion Tool

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 the AWS Schema Conversion Tool (p. 108).
   - 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 the AWS Schema Conversion Tool (p. 115).

Collecting or Uploading Statistics for the AWS Schema Conversion Tool

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.
Later, when you review the suggested keys, if you are not satisfied with the results, you can collect additional statistics and repeat this procedure. For more information, see Managing and Customizing Keys in the AWS Schema Conversion Tool (p. 115).

Creating Mapping Rules in the AWS Schema Conversion Tool

Before you convert your schema with the AWS Schema Conversion Tool, you can set up rules that change the data type of columns, move objects from one schema to another, and change the names of objects. For example, if you have a set of tables in your source schema named \texttt{test\_TABLE\_NAME}, you can set up a rule that changes the prefix \texttt{test\_} to the prefix \texttt{demo\_} in the target schema.

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

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

You can create mapping rules for the following objects:
- Database
- Schema
- Table
- Column

Creating Mapping Rules

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

To create mapping rules


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

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

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

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

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

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

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

Viewing Mapping Rules for Objects

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

Exporting Mapping Rules

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

To export mapping rules

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

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

Converting Your Schema by Using the AWS Schema Conversion Tool

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

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

**Converting Schema**

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

After you have converted the schema from your source database, you can choose schema items from the left panel of your project and view the converted schema in the center panels of your project. The lower-center panel displays the properties of and the SQL command to create the converted schema, as shown following.
After you have converted your schema, you can save your project. The schema information from your source database is saved with your project. This functionality means that you can work offline without being connected to your source database. AWS SCT connects to your source database to update the schema in your project if you choose Refresh from Database for your source database. For more information, see Updating and Refreshing Your Converted Schema in the AWS Schema Conversion Tool (p. 117).

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 Assessment Reports with AWS Schema Conversion Tool (p. 77).

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. 119).

**Editing Converted Schema**

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

**To edit converted schema**

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

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

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

### Clearing a Converted Schema

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

After you convert your schema with the AWS Schema Conversion Tool, 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 the AWS Schema Conversion Tool (p. 107).

Handling Manual Conversions in the AWS Schema Conversion Tool

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

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

- Modify your source database schema.
- Modify your target database schema.

Modifying Your Source Schema

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

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

Modifying Your Target Schema

For some items, it might be easier to apply the converted schema to your target database, and then add equivalent schema items manually to your target database for the items that couldn't be converted automatically. You can write all of the schema that can be converted automatically to your target database by applying the schema. For more information, see Saving and Applying Your Converted Schema in the AWS Schema Conversion Tool (p. 118).

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

Warning

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

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

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

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

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

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

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

When the AWS Schema Conversion Tool generates converted schema (as shown in Converting Your Schema by Using the AWS Schema Conversion Tool (p. 111)), 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 Assessment Reports with AWS Schema Conversion Tool (p. 77).

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

Saving Your Converted Schema to a File

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

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

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

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

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

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

- Greenplum: AWS_GREENPLUM_EXT
- Microsoft SQL Server: AWS_SQLSERVER_EXT
- Netezza: AWS_NETEZZA_EXT
- Oracle: AWS_ORACLE_EXT
- Teradata: AWS_TERADATA_EXT
- Vertica: AWS_VERTICA_EXT

For more information, see Using the AWS Schema Conversion Tool Extension Pack (p. 186).
Python Libraries

To create custom functions in Amazon Redshift, you use the Python language. Use the AWS SCT extension pack to install python libraries for your Amazon Redshift database. For more information, see Using the AWS Schema Conversion Tool Extension Pack (p. 186).

Optimizing Amazon Redshift by Using the AWS Schema Conversion Tool

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 ETL Processes to AWS Glue

In addition to migrating your schema and data using AWS SCT, you can also migrate extraction, transformation, and load (ETL) processes. This type of migration includes the conversion of ETL-related business logic located either inside the source data warehouses or in external scripts that are run separately. After migration, the ETL processes run in AWS Glue. You run the ETL to AWS Glue migration as a separate project from converting your data definition language (DDL) statements and data.
Currently, only Oracle ETL and Teradata BTEQ conversions to AWS Glue are supported.

**Topics**
- Prerequisites (p. 124)
- Understanding the AWS Glue Data Catalog (p. 124)
- Limitations for Converting with AWS Glue (p. 124)
- Converting Using AWS Glue in the AWS SCT UI (p. 125)
- Converting Using the Python API for AWS Glue (p. 130)

**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, you must 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 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>
<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 DPUs used by a development endpoint at one time</td>
<td>5</td>
</tr>
<tr>
<td>Maximum data processing units (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></td>
<td>For compatibility with other metadata stores, such as Apache Hive, the name is changed to use lowercase characters.</td>
</tr>
<tr>
<td>Connection name length</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Crawler name length</td>
<td>Unlimited</td>
</tr>
</tbody>
</table>

**Converting Using AWS Glue in the AWS SCT UI**

In the following sections, you can find an outline of the process to follow to convert ETL using 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.
Step 1: Create a New Project

First, start a new project. To do this, start AWS SCT and choose File, New Project. Provide a name and a location to save your new project.

Choose Data warehouse (OLAP), and then choose your source and target database engines.

Make sure that you have enabled Use AWS Glue in the project settings. To view project settings, choose Settings, Project Settings.
To finish preparing to import your ETL, establish connections to your source and target database engines. Choose the menu items that begin with the words **Connect to**. These labels are customized to show the database engines that you chose when you created the project. For example, if you chose Oracle and Amazon Redshift, the menu displays the options **Connect to Oracle** and **Connect to Amazon Redshift**.

After you choose the option to connect to your source database, enter your connection information. You know that the connection is complete when SCT displays a list of your database objects.

After you choose the option to connect to your target database, enter your connection information. Next, choose the **AWS Glue** tab that appears. 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. The following screenshot shows an example of what this screen looks like. For the example, we're using Amazon Redshift as our target database.
**Note**

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.

AWS Glue creates a database on the source database server and also on the target database server to help with the ETL conversion. The database on the target server contains the AWS Glue Data Catalog. To locate specific objects, use search on the source or target panels.

To see how a specific object converts, locate an item you want to convert, and choose **Convert schema** from its context (right-click) menu. AWS SCT transforms it into a script.
When the transformation is complete, the script is stored on the target server in the Scripts folder of the AWS Glue Data Catalog. You can view this on your source database pane on the right side of the screen. To upload the script to 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 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.

On the first tab, 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.

On the second step, or tab, of the wizard, you can name your job, and directly configure settings for AWS Glue. On this screen, you can configure the following settings:

- 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
- Dependent .jars 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

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/.

You can test the new job to make sure that it’s working correctly. To do so, first check the data in your source table, then verify that the target table is empty. Run the job, and check again. You can view error logs from the AWS Glue console.

### Converting Using the Python API for AWS Glue

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.apq1mpqso.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.

```python
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: Creating 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 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, you must 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: `abcde03_dev_ora_glue`
- Use an IAM role that already exists. For more information on creating IAM roles, see Creating IAM Roles in the IAM User Guide.
- Use the name of the database that you created in the previous steps.
• 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'|'DEPRECIATE_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'
)
```

Because we're using Amazon Redshift as our target for this example, Amazon Redshift data types map to Glue data type 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>smalint</td>
<td>smallint</td>
</tr>
<tr>
<td>Type</td>
<td>Corresponding Type</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------</td>
</tr>
<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>timestamp tz</td>
<td>timestamp</td>
</tr>
</tbody>
</table>
Using the AWS Schema Conversion Tool with the AWS Database Migration Service

Using an AWS SCT Replication Agent with AWS DMS

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

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

For more information, see Using Data Extraction Agents (p. 135).

Using an AWS SCT Data Extraction Agent with AWS DMS

AWS SCT provides a data extraction agent to facilitate migrations from Apache Cassandra to Amazon DynamoDB. Cassandra and DynamoDB are NoSQL databases, but they differ in system architecture and data representation. AWS SCT provides wizard-based workflows for automating the Cassandra-to-DynamoDB migration process, and integrates with AWS Database Migration Service (AWS DMS) to perform the actual migration.

For more information, see Using Data Extraction Agents (p. 135).
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 SCT is extensible, so that you can address these scenarios using an agent—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.

The following topics describe these scenarios in more detail, and explain how to use AWS SCT agents during the migration process.

Topics
- Migrating Data From an On-Premises Data Warehouse to Amazon Redshift (p. 135)
- Migrating Data From Apache Cassandra to Amazon DynamoDB (p. 159)

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 an AWS Snowball device. You can then use AWS SCT to copy the data to Amazon Redshift.

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

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

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

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

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

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

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

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

**Topics**

- Prerequisite Settings for Amazon S3 and Security for Data Extraction Agents (p. 137)
- Installing Extraction Agents (p. 138)
- Registering Extraction Agents with the AWS Schema Conversion Tool (p. 141)
- Hiding and Recovering Information for an AWS SCT Agent (p. 142)
- Creating Data Extraction Filters in the AWS Schema Conversion Tool (p. 143)
Prerequisite Settings for Amazon S3 and Security for Data Extraction Agents

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

Amazon S3 Settings

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

Security Settings

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

To set up secure communication with your extraction agent

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

   Choose the Security tab as shown following.
3. Choose **Generate Trust and Key Store**, or choose **Select existing Trust and Key Store**.

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

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

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

**Installing Extraction Agents**

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

Extraction agents are currently supported on the following operating systems:

- macOS
- Microsoft Windows
- Red Hat Enterprise Linux (RHEL) 6.0
- Ubuntu Linux (version 14.04 and later)
Use the following procedure to install extraction agents. Repeat this procedure for each computer that you want to install an extraction agent on.

To install an extraction agent

1. If you have not already downloaded the AWS SCT installer file, follow the instructions at Installing, Verifying, and Updating the AWS Schema Conversion Tool (p. 4) to download it. The .zip file that contains the AWS SCT installer file also contains the extraction agent installer file.

2. Locate the installer file for your extraction agent in a subfolder named agents. For each computer operating system, the correct file to install the extraction agent is shown following.

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

3. To install the extraction agent on a separate computer, copy the installer file to the new computer.

4. Run the installer file. Use the instructions for your operating system, shown following.

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

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

6. Copy the SSL trust and key stores (.zip or individual files) that you generated in an earlier procedure. If you copy the .zip file to a new computer, extract the individual files from the .zip file on the new computer.
You can put the files anywhere you want. However, note the locations because in a later procedure you tell the agent where to find the files.

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

**Configuring Extraction Agents**

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

**To configure your extraction agent**

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

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

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

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

```
port=8888
vendor=ORACLE
driver.jars=<driver path>/Install/Drivers/ojdbc7.jar
location=<output path>/dmt/8888/out
extractor.log.folder=<log path>/dmt/8888/log
extractor.storage.folder=<storage path>/dmt/8888/storage
extractor.start.fetch.size=20000
extractor.out.file.size=10485760
ssl.option=OFF
#ssl.option=ON
#ssl.keystore.path=<key store path>/dmt/8888/vault/keystore
#ssl.truststore.path=<trust store path>/dmt/8888/vault/truststore
```

**Starting Extraction Agents**

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

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

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

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Start Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>macOS</td>
<td>Run the <code>StartAgent.command</code> file.</td>
</tr>
<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**. The **Agents** tab appears. If you have previously registered agents, they appear in a grid at the top of the tab as shown following.
3. Choose Register. The **New Agent Registration** dialog box appears.

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

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

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

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

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

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

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

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

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

Sorting Data Before Migrating Using AWS SCT

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

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

To sort data before migrating using AWS SCT

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

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.

If you enabled logging when you set up the task, you can view the log:

1. **Choose Download Log**. A message appears with the name of the folder that contains the log file. Dismiss the message.
2. A link appears in the **Task details** tab. Choose the link to open the folder that contains the log file.

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

## Data Extraction Using a AWS Snowball Device

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

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

The following steps need to occur to migrate data from a local data store to an AWS data store using AWS Snowball.

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

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

The following sections provide detailed information on the migration steps.

**Step 1: Create an AWS Snowball Job**

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

**Step 2: Unlock the AWS Snowball Edge Device**

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

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

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

**Step 3: Create a New AWS SCT Project**

Next, you create a new AWS SCT project.

**To create a new project in AWS SCT**

1. Start AWS SCT, and choose **New Project** for **File**. The **New 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>Project Name</strong></td>
<td>Type a name for your project, which is stored locally on your computer.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>Type the location for your local project file.</td>
</tr>
<tr>
<td><strong>OLTP</strong></td>
<td>Choose <strong>Transactional Database (OLTP)</strong>.</td>
</tr>
<tr>
<td><strong>Source DB Engine</strong></td>
<td>Choose your source data store.</td>
</tr>
<tr>
<td><strong>Target DB Engine</strong></td>
<td>Choose your target data store.</td>
</tr>
</tbody>
</table>
3. Choose **OK** to create your AWS SCT project.
4. (Optional) Test your connection.

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

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

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

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

**To install on Oracle**

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

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

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

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

```
vi cat <product_dir>/bin/site_arep_login.sh
```

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

**To install on Microsoft SQL Server**

Install the Microsoft ODBC Driver

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...
To install on SAP Sybase

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

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

```bash
export SYBASE_HOME=/opt/sap
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SYBASE_HOME/
    DataAccess64/ODBC/lib:$SYBASE_HOME/DataAccess/ODBC/
    lib:$SYBASE_HOME/OCS-16_0/lib:$SYBASE_HOME/OCS-16_0/
    lib3p64:$SYBASE_HOME/OCS-16_0/lib3p
```

The /etc/odbcinst.ini should include these entries

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

To install on MySQL

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

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

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

To install on PostgreSQL

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

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

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

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

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

```ini
[PostgreSQL]
Description = PostgreSQL ODBC driver
Driver = /usr/pgsql-9.4/lib/psqlodbc.so
```
Setup = /usr/pgsql-9.4/lib/psqlodbcw.so
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 & DMS Task

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

To create the end-to-end migration task

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

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

3. Add your task information.

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Name</td>
<td>Type a name for the task.</td>
</tr>
<tr>
<td>Agent</td>
<td>Choose DMS Agent.</td>
</tr>
</tbody>
</table>
For This Parameter | Do This
--- | ---
Replication Instance | Choose the AWS DMS replication instance that you want to use.
Migration Type | Choose the type of migration you want.
Choose **Migrate existing data** to migrate the contents of the chosen schema. This process is called a full load in AWS DMS.
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.
Target table preparation mode | Choose the preparation mode you want to use.
**Truncate** - Tables are truncated without affecting table metadata.
**Drop tables on target** - The tables are dropped and new tables are created in their place.
**Do nothing** - Data and metadata of the target tables are not changed.
IAM role | Choose the predefined IAM role that has permissions to access the Amazon S3 bucket and the target database. For more information about the permissions required to access an Amazon S3 bucket, see Amazon S3 Settings (p. 137).
Logging | Choose **Enable** to have AWS CloudWatch create logs for the migration. You incur charges for this service. For more information about AWS CloudWatch, see How Amazon CloudWatch Works.
Description | Type a description of the task.
Use Snowball | Choose this check box to use Snowball.
Job Name | Choose the AWS Snowball job name you created.
Snowball IP | Type the IP address of the AWS Snowball appliance.
Port | Type the port value for the AWS Snowball appliance.
Local AWS S3 Access key | Type the AWS access key for the account you are using for the migration.
Local AWS S3 Secret key | Type the AWS secret key for the account you are using for the migration.

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

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

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

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

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 Connecting to Clusters From Client Tools and Code.</td>
</tr>
<tr>
<td>Extract and Upload</td>
<td>The extraction agents saved your data as files in your Amazon S3 bucket. You can use the Amazon Redshift COPY command to load your data to Amazon Redshift. For more information, see Loading Data from Amazon S3 in the Amazon Redshift documentation. There are multiple folders in your Amazon S3 bucket, corresponding to the extraction tasks that you set up. When you load your data to Amazon Redshift, specify the name of the manifest file created by each task. The manifest file appears in the task folder in your S3 bucket as shown following.</td>
</tr>
</tbody>
</table>
## Using Virtual Partitioning with AWS Schema Conversion Tool

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

- The RANGE partition type works with numeric and date and time data types.
- The LIST partition type works with numeric, character, and date and time data types.
- The DATE AUTO SPLIT partition type works with date and time data types.

### 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.

<table>
<thead>
<tr>
<th>Migration Mode</th>
<th>Data Location</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Amazon S3</td>
</tr>
<tr>
<td></td>
<td>extractortest</td>
</tr>
</tbody>
</table>

- dms_sample_person_chunk_10.csv.lzo
- dms_sample_person_chunk_9.csv.lzo
- unit.manifest

### Using Virtual Partitioning

Using Virtual Partitioning with AWS Schema Conversion Tool (AWS SCT) allows you to create virtual partitions of your data. This can be particularly useful for managing large non-partitioned tables. AWS SCT supports three partition types:

1. **RANGE Partition Type**: Works with numeric and date and time data types.
2. **LIST Partition Type**: Works with numeric, character, and date and time data types.
3. **DATE AUTO SPLIT Partition Type**: Works with date and time data types.

After extracting your data, you can manually copy it to your Amazon S3 bucket and proceed with the instructions for Extract and Upload.
AWS SCT validates the values you provide for creating a partition. For example, if you attempt to partition a column with data type NUMERIC but you provide values of a different data type, AWS SCT throws an error.

**Limits When Creating Virtual Partitioning**

These are limitations to creating a virtual partition:

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

**RANGE Partition Type**

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

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

<table>
<thead>
<tr>
<th>Partition 1: WHERE LO_TAX &lt;= 10000.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 2: WHERE LO_TAX &gt; 10000.9 AND LO_TAX &lt;= 15005.5</td>
</tr>
<tr>
<td>Partition 3: WHERE LO_TAX &gt; 15005.5 AND LO_TAX &lt;= 25005.95</td>
</tr>
</tbody>
</table>

**To create a RANGE virtual partition**

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition Type</td>
<td>Choose <strong>RANGE</strong>. 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 <strong>New Value</strong> box, then choosing the plus sign to add the value.</td>
</tr>
<tr>
<td>Load From File</td>
<td><em>(Optional)</em> Type the name of a .csv file that contains the partition values.</td>
</tr>
</tbody>
</table>

5. Choose *OK*.

**LIST Partition Type**

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

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

<table>
<thead>
<tr>
<th>Partition</th>
<th>WHERE LO_ORDERKEY = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition2</td>
<td>WHERE LO_ORDERKEY = 2</td>
</tr>
<tr>
<td>Partition3</td>
<td>WHERE LO_ORDERKEY = 3</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>PartitionN</td>
<td>WHERE LO_ORDERKEY = USER_VALUE_N</td>
</tr>
</tbody>
</table>

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

**To create a LIST virtual partition**

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

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

**DATE AUTO SPLIT Partition Type**

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

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

<table>
<thead>
<tr>
<th>Partition</th>
<th>WHERE LO_ORDERDATE &gt;= '1954-10-10' AND LO_ORDERDATE &lt; '1954-10-24'</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition2</td>
<td>WHERE LO_ORDERDATE &gt;= '1954-10-24' AND LO_ORDERDATE &lt; '1954-11-06'</td>
</tr>
<tr>
<td>Partition3</td>
<td>WHERE LO_ORDERDATE &gt;= '1954-11-06' AND LO_ORDERDATE &lt; '1954-11-20'</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>PartitionN</td>
<td>WHERE LO_ORDERDATE &gt;= USER_VALUE_N AND LO_ORDERDATE &lt;= '2017-08-13'</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition Type</td>
<td>Choose <strong>DATE AUTO SPLIT</strong>. 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>Type the interval unit, and choose the value for that unit.</td>
</tr>
</tbody>
</table>

5. Choose **OK**.

---

### Migrating LOBs to Amazon Redshift

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

**To migrate LOBs to Amazon Redshift**

1. Open an AWS SCT project.
2. For **Actions**, choose **Create Local Task**.
3. Choose the **Advanced** tab.
4. For **S3 bucket LOBs folder**, type the name of the folder in an S3 bucket where you want the LOBs stored.
5. 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>
<tr>
<td>Contention delays</td>
<td>Avoid having too many agents accessing your data warehouse at the same time.</td>
</tr>
<tr>
<td>An agent goes down temporarily</td>
<td>If an agent is down, the status of each of its tasks appears as failed in AWS SCT. If you wait, in some cases the agent can recover. In this case, the status of its tasks updates in AWS SCT.</td>
</tr>
</tbody>
</table>
### Issue

An agent goes down permanently

### Troubleshooting Suggestions

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

- Install an agent on a new computer.
- Configure the new agent with the same settings, including port number and working folder, as the original agent.
- Start the agent. After the agent starts, the task discovers the new available agent and continues running on the new agent.

---

**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 datacenter*—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 datacenter and writes it to an Amazon S3 bucket. AWS SCT then reads the data from Amazon S3 and writes it to Amazon 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 **datacenter** 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 Amazon DynamoDB:

**Topics**
- Prerequisites for Migrating From Cassandra to DynamoDB (p. 161)
- Create a New AWS SCT Project (p. 163)
- Create a Clone Datacenter (p. 165)
- Install, Configure and Run the Data Extraction Agent (p. 172)
- Migrate Data From the Clone Datacenter to Amazon DynamoDB (p. 175)
- Post-Migration Activities (p. 179)

**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
- 2.2
- 2.1.20 or later

Other versions of Cassandra aren't supported.

**Amazon S3 Settings**

When the AWS SCT data extraction agent runs, it reads data from your clone datacenter 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 Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13).

**Amazon EC2 Instance for Clone Datacenter**

As part of the migration process, you'll need to create a clone of an existing Cassandra datacenter. 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 datacenter independently of your existing Cassandra datacenter.

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/](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:

   ![Security Settings](image)

   - **Generate Trust and Key Store**
   - **Select existing Trust and Key Store**

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.

5. After you have specified the trust store and key store, choose **OK** to close the **Global Settings** dialog box.
Create a New AWS SCT Project

After you have performed the steps in Prerequisites for Migrating From Cassandra to DynamoDB (p. 161), you’re ready to create a new AWS SCT project for your migration. Follow these steps:

1. From the File menu, choose New project.

Add the following information:

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>Type a name for your project, which is stored locally on your computer.</td>
</tr>
<tr>
<td>Location</td>
<td>Type the location for your local project file.</td>
</tr>
<tr>
<td>NoSQL database</td>
<td>Choose NoSQL database.</td>
</tr>
<tr>
<td>Source database engine</td>
<td>Choose Cassandra.</td>
</tr>
<tr>
<td>Target database engine</td>
<td>Choose Amazon DynamoDB.</td>
</tr>
</tbody>
</table>
Choose **OK** to create your AWS SCT project.

2. From the menu bar, choose **Connect to Cassandra**.

Add the following information:

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server name</strong></td>
<td>Type the host name or IP address for your Cassandra cluster.</td>
</tr>
<tr>
<td><strong>Server port</strong></td>
<td>Type the port number where Cassandra is listening for connection requests. For example: <strong>9042</strong></td>
</tr>
<tr>
<td><strong>User name</strong></td>
<td>Type a valid user name for connecting to the Cassandra cluster.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Type the password that's associated with the user name.</td>
</tr>
</tbody>
</table>

3. Choose **OK**. AWS SCT tests the connection to ensure that it can access your Cassandra cluster.

4. From the menu bar, choose **Connect to Amazon DynamoDB**.
Add the following information:

<table>
<thead>
<tr>
<th>For This Parameter</th>
<th>Do This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy from AWS profile</td>
<td>If you’ve already set up your AWS credentials, choose the name of an existing profile.</td>
</tr>
<tr>
<td>AWS access key</td>
<td>Type your AWS access key.</td>
</tr>
<tr>
<td>AWS secret key</td>
<td>Type the secret key associated with your AWS access key.</td>
</tr>
<tr>
<td>Region</td>
<td>Choose an AWS region. AWS SCT will migrate your data to Amazon DynamoDB in that region.</td>
</tr>
</tbody>
</table>
1. In the AWS SCT window, on the left-hand side (source), expand the **Datacenters** node and choose one of your existing Cassandra datacenters.

2. From the **Actions** menu, choose **Clone Datacenter for Extract**.
Create a Clone Datacenter

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 you can extract data from the original Cassandra Cluster, the original Cassandra Cluster will be reconfigured to extract data from the original Cluster using the Repl.

SCT will perform data extraction from a new cluster.

This wizard automates the preparatory steps necessary to perform data extraction.

This wizard connects to the source Cassandra Cluster to discover and configure files for each node.

The diagram below provides an overview of the Clone Datacenter process.
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>Type 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>Type 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>
<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>Type the JMX username 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 datacenter, it will be named similarly to the source datacenter, but with the suffix your provide. For example, if the source datacenter 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:
### Target Cassandra datacenter

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hints directory</td>
<td>/var/lib/cassandra/hints</td>
</tr>
<tr>
<td>Data file directory</td>
<td>/var/lib/cassandra/data</td>
</tr>
<tr>
<td>Commitlog directory</td>
<td>/var/lib/cassandra/commitlog</td>
</tr>
<tr>
<td>CDC raw directory</td>
<td>/var/lib/cassandra/cdc</td>
</tr>
<tr>
<td>Snitch mode</td>
<td>$2Snitch</td>
</tr>
<tr>
<td>Datacenter suffix</td>
<td>_tgt</td>
</tr>
</tbody>
</table>

Specify target node: **Add new node** or **Select existing node**.

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 Datacenter (p. 161)*.

When the settings are as you want them, choose **Add**. The node appears in the list:
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 policy for using the current datacenter.
Also, use LOCAL_QUORUM/LOCAL

SCT will now verify network compatibility between source and target. Provide the parameters for the target datacenter configuration process. You can also click "Show Log".

**Target Cassandra datacenter**

- Hints directory: /var/lib/cassandra/hints
- Data file directory: /var/lib/cassandra/dt
- Commitlog directory: /var/lib/cassandra/commit
- CDC raw directory: /var/lib/cassandra/cdc
- Snitch mode: ec2Snitch
- Datacenter suffix: _tgt

**List of nodes**

<table>
<thead>
<tr>
<th>Private IP</th>
<th>Private port</th>
<th>Public IP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

Show Cassandra configuration
10. Choose **Next** to continue. The following confirmation box appears:

![Confirmation](image)

Choose **OK** to continue. AWS SCT reboots your source datacenter, 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 datacenter. If your cluster is running Cassandra version 3, then you can choose which keyspace(s) you want to copy to the clone datacenter.

12. When you are ready to begin replicating data to your clone datacenter, 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 datacenter. 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 datacenter, 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 the AWS Schema Conversion Tool (p. 4)).

**Note**  
We recommend that you run the agent on an Amazon EC2 instance. The 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 -version
```

3. Install the sshfs package:

```
sudo yum install sshfs
```

4. Install the expect package:

```
sudo yum install expect
```

5. Edit the `/etc/fuse.conf` file, and uncomment the string `user_allow_other`:

```
# 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 the AWS Schema Conversion Tool (p. 4)). 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>Filename</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:

   ```
sudo dpkg -i aws-cassandra-extractor-n.n.n.deb
   ```

   • For CentOS:

   ```
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.
   • `/mnt/cassandra-data-extractor/`—for mounting home and data folders.
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. 162).) 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:

```bash
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
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:
```

```
version 1.0
174
```
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 Datacenter to Amazon DynamoDB**

You are now ready to perform the migration from the clone datacenter 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:

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>Description</td>
<td>Type a short description for this agent.</td>
</tr>
<tr>
<td>Host name</td>
<td>Type the hostname of the Amazon EC2 instance you used for Install, Configure and Run the Data Extraction Agent (p. 172)</td>
</tr>
<tr>
<td>Port</td>
<td>Type the port number for the agent. (The default port number is 8080.)</td>
</tr>
<tr>
<td>Password</td>
<td>If you are using SSL, leave this field blank; otherwise, type the password for logging into the host.</td>
</tr>
<tr>
<td>Use SSL</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>Trust store</td>
<td>Choose the trust store you configured in Install, Configure and Run the Data Extraction Agent (p. 172).</td>
</tr>
<tr>
<td>Key store</td>
<td>Choose the key store you configured in Install, Configure and Run the Data Extraction Agent (p. 172).</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 datacenter that you created in Create a Clone Datacenter (p. 165).

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>Task name</td>
<td>Type a short name for the AWS DMS task to be created.</td>
</tr>
<tr>
<td>Replication instance</td>
<td>Choose the AWS DMS replication instance that you want to use.</td>
</tr>
</tbody>
</table>
### For This Parameter | Do This
--- | ---
**Migration type** | Choose **Migrate existing data and replication ongoing changes**. This will migrate the tables in your Cassandra clone datacenter to DynamoDB, and then capture all ongoing changes. This process is called full load and CDC in AWS DMS.

**Target table preparation mode** | If you already have corresponding tables in DynamoDB and want to delete them prior to migration, choose **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 When the AWS SCT data extraction agent runs, it reads data from your clone datacenter 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 Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13). (p. 161).

**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** | Type 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 screen shot:
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 the AWS Schema Conversion Tool

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

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

Overview of Converting Application SQL

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

- **Create an application conversion project** – The application conversion project is a child of the database schema conversion project. Each database schema conversion project can have one or more child application conversion projects. For more information, see Creating Application Conversion Projects in the AWS Schema Conversion Tool (p. 180).

- **Analyze and convert your SQL code** – AWS SCT analyzes your application, extracts the SQL code, and creates a local version of the converted SQL for you to review and edit. The tool doesn't change the code in your application until you are ready. For more information, see Analyzing and Converting Your SQL Code by Using the AWS Schema Conversion Tool (p. 183).

- **Create an application assessment report** – The application assessment report provides important information about the conversion of the application SQL code from your source database schema to your target database schema. For more information, see Creating and Using the Assessment Report (p. 184).

- **Edit, apply changes to, and save your converted SQL code** – The assessment report includes a list of SQL code items that can't be converted automatically. For these items, you can edit the SQL code manually to perform the conversion. For more information, see Editing and Saving Your Converted SQL Code with the AWS Schema Conversion Tool (p. 185).

Creating Application Conversion Projects in the AWS Schema Conversion Tool

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 an application conversion project.

**To create an application conversion project**

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

2. Add the following project information.

<table>
<thead>
<tr>
<th><strong>For This Parameter</strong></th>
<th><strong>Do This</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Project Name</td>
<td>Type a name for your application conversion project. Each database schema conversion project can have one or more child application conversion projects, so choose a name that makes sense if you add more projects later.</td>
</tr>
<tr>
<td>Location</td>
<td>Type the location of the source code for your application.</td>
</tr>
<tr>
<td>Project language</td>
<td>Choose one of the following:</td>
</tr>
<tr>
<td></td>
<td>• JAVA</td>
</tr>
<tr>
<td></td>
<td>• C++</td>
</tr>
<tr>
<td>For This Parameter</td>
<td>Do This</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>• C#</td>
</tr>
<tr>
<td></td>
<td>• Any</td>
</tr>
<tr>
<td>SQL parameter style for target database</td>
<td>Choose one of the following:</td>
</tr>
<tr>
<td></td>
<td>• Same as in source</td>
</tr>
<tr>
<td></td>
<td>• Positional (?)</td>
</tr>
<tr>
<td></td>
<td>• Indexed (:1)</td>
</tr>
<tr>
<td></td>
<td>• Indexed ($1)</td>
</tr>
<tr>
<td></td>
<td>• Named (@name)</td>
</tr>
<tr>
<td></td>
<td>• Named (:name)</td>
</tr>
<tr>
<td>Select source database schema</td>
<td>In the source tree, choose the schema used by your application code.</td>
</tr>
</tbody>
</table>

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

The project window opens.

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

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

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

### To analyze and convert your SQL code

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

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

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

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

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

Creating an Application Assessment Report

Use the following procedure to create an application assessment report.

To create an application assessment report

1. In the application conversion project window, choose Create Report from the Actions menu.
   
   The report is created and opens in the application conversion project window.

2. Review the Summary tab.

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

3. Choose the SQL Conversion Actions tab and review the information.
The **SQL Conversion Actions** tab contains a list of SQL code items that can't be converted automatically. There are also recommendations for how to manually convert the SQL code. You edit your converted SQL code in a later step. For more information, see *Editing and Saving Your Converted SQL Code with the AWS Schema Conversion Tool* (p. 185).

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

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

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

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

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

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

**Important**

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

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

The AWS SCT Extension Pack includes the following components:

- **DB schema** – Includes SQL functions, procedures, and tables for emulating some OLTP and OLAP objects (for example, sequence) or unsupported built-in-functions from the source database. This schema is named in the format `aws_<database engine name>_ext`.
- **Custom Python library (for select OLAP databases)** – Includes a set of Python functions that emulate unsupported built-in database functions. Use this library when you migrate from one of the supported databases to Amazon Redshift.
  
  For more information on this library, see Using the Custom Python Library for the AWS SCT Extension Pack (p. 187).
- **AWS Lambda functions (for select OLTP databases)** – Includes AWS Lambda functions that emulate complex database functionality, such as job scheduling and sending emails.

The following sections discuss the AWS SCT Extension Pack.

**Topics**
- Using the Extension Pack Schema (p. 186)
- Using the Custom Python Library for the AWS SCT Extension Pack (p. 187)
- Using the AWS Lambda Functions from the AWS SCT Extension Pack (p. 188)

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

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

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

### Using the Extension Pack Schema

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

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

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

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

- Greenplum: AWS_GREENPLUM_EXT
- Microsoft SQL Server: AWS_SQLSERVER_EXT
- Netezza: AWS_NETEZZA_EXT
- Oracle: AWS_ORACLE_EXT
- Teradata: AWS_TERADATA_EXT
- Vertica: AWS_VERTICA_EXT

Using the Custom Python Library for the AWS SCT Extension Pack

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

If you are converting a transactional database, instead see Using the AWS Lambda Functions from the AWS SCT Extension Pack (p. 188).

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

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

Using AWS Services to Upload Custom Python Libraries

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

Applying the Extension Pack

Use the following procedure to apply the extension pack.

To apply the extension pack

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

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

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

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

When you are done, choose Next.


When you are done, choose Finish.

Using the AWS Lambda Functions from the AWS SCT Extension Pack

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

Using AWS Lambda Functions to Emulate Database Functionality

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

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

Applying the Extension Pack

Use the following procedure to apply the extension pack.

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

To apply the extension pack

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

   The extension pack wizard appears.

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

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

4. On the Email Sending Service page, do the following:
   - If you are reinstalling the extension pack schema only, choose Skip this step for now, and then choose Next.
   - If you are installing AWS services and you have an existing Lambda function, you can provide it. Otherwise, the wizard creates it for you. When you are done, choose Next.

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

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

General Memory Management and Performance Options

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

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

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

Configuring Additional Memory

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

**To modify the amount of memory AWS SCT consumes**

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

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

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

Cannot load objects from an Oracle source database

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

- 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_privs and grants the user with this ID the minimum permissions required to convert schema from an Oracle source database.

```
create user min_privs identified by min_privs;
grant connect to min_privs;
grant select_catalog_role to min_privs;
grant select any dictionary to min_privs;
```
## Release Notes for the AWS Schema Conversion Tool

This section contains release notes for AWS Schema Conversion Tool, starting with version 1.0.611.

### Release Notes for the AWS Schema Conversion Tool Build 632

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.632.

<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>SCT UI - Added new tab to show errors that happen when applying scripts.</td>
</tr>
<tr>
<td>SAP-ASE</td>
<td>ALL</td>
<td>You can now save the source tree as SQL.</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL or Aurora</td>
<td>• Now converts empty string literals to <strong>NULL</strong> everywhere: object metadata, parsable statements, scripts, SQL statements in application source code etc.</td>
</tr>
<tr>
<td></td>
<td>PostgreSQL</td>
<td>• REPLACE now has same behavior in Oracle and PostgreSQL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Now accepts more then 100 arguments, that were not allowed for functions and procedures by default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions <code>aws_oracle_ext.instr</code> no longer returns wrong result.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions LEAST and GREATEST now work the same.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Extpack function <code>immutable_concat_ws</code> should no longer be used instead of <code>standard_concat_ws</code> in the views.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Implemented immutable concat extpack function.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Public synonym now longer expanded in views.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for <code>dbms_application_info.set_module</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Now supporting previously incompatible number type variables for Error Codes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Now supporting <code>FETCH BULK COLLECT INTO</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Now supporting <code>SAVEPOINT</code>, <code>ROLLBACK TO SAVEPOINT</code> inside of functions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Now supporting cursor with <code>SELECT...INTO...</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Now supporting package's collection variable as a <code>DEFAULT</code> value of input parameter of routine.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A parameter or variable's type declaration is based on a system's view column <code>%TYPE</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Now supporting global nested tables of <code>%ROWTYPE</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Support for packages with subtypes of <code>%TYPE</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fix: Insert into collection element field caused transformer error.</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>
</tbody>
</table>
| SAP ASE             | PostgreSQL or Aurora PostgreSQL | • Now supporting correct conversion of multi-table UPDATE statement with unresolved temporary table.  
• Correct object resolves, when the schema name is not specified or only the database name is used.  
• Now supporting CONVERT function (Extension pack).                                                                                   |
| Netezza             | Redshift                    | • Stored code. Transaction control improved support (COMMIT/ ROLLBACK).  
• Change severity "critical" to "high" for unresolved AI 15028. Added operator /=/ processing in queries.  
• Decrease "Estimated Time to Resolve Manually" for AIs 15001, 15002, 15003.  
• Conversion of /=/ operator in sql statements.  
• Datatype mapping for quoted datatypes (e.g. "VARCHAR", "TIME", and so on) improvements.                                                                 |
| Amazon RDS for Oracle | Redshift                    | Fix: For copy command error that occurred when loading to Redshift from RDS Oracle using SCT Data extraction agent on any Japanese environment OS. Added command "ALTER SESSION SET NLS_DATE_LANGUAGE = 'ENGLISH'" for Oracle before extracting data. |

Resolved:

• FIX: Typo in Current project settings menu  
• FIX: Unable to convert to upper case with the mapping rule (when PostgreSQL was the target)
Release Notes for the AWS Schema Conversion Tool Build 631

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.631.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle and OracleDW</td>
<td>All</td>
<td>Added support for count mechanism and SQL code lines count</td>
</tr>
<tr>
<td>Oracle</td>
<td>MariaDB (10.2/10.3/10.3 SQL MODE=ORACLE)</td>
<td>Hierarchical queries using Oracle CONNECT BY syntax are transferred to Common Table Expressions (WITH RECURSIVE)</td>
</tr>
<tr>
<td>Oracle</td>
<td>MySQL</td>
<td>Fixed complex join with the table appearing on both sides of the join</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>MySQL</td>
<td>Support for CONVERT function</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>MySQL (Aurora)</td>
<td>FIX: Conversion of INSERT with multi-table SELECT-implementation and conversion of multi-table UPDATE statement with a temporary table-implementation</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>MySQL</td>
<td>FIX: incorrect schema and object definition when you call an object and don’t specify a schema</td>
</tr>
<tr>
<td></td>
<td>MySQL (Aurora)</td>
<td>System objects without schemas now convert as is.</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>MySQL</td>
<td>Fixed: incorrect conversion of INSERT with multi-table SELECT</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>PostgreSQL</td>
<td>Comment in try/catch block is now transferred to target</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improved support for stored procedure parameters with defaults</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIX: Removing entire code from proc when ALTER TABLE and ENABLE/DISABLE TRIGGER ALL are used</td>
</tr>
<tr>
<td>SAP ASE</td>
<td>PostgreSQL</td>
<td>Support CONVERT function</td>
</tr>
<tr>
<td>Oracle</td>
<td>PostgreSQL</td>
<td>Support for ProC to ECPG code conversion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improved support for RAWTOHEX/HEXTORAW/UTL_RAW conversion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improved support for global cursors, to allow OPEN and FETCH commands in separate stored routines. With this improvement, the cursor can be opened in one procedure, fetched in another procedure, and closed in a third procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Option to switch between procedure and function for PostgreSQL</td>
</tr>
</tbody>
</table>
### What's new, enhanced, or fixed

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBMS_XMLGEN</strong></td>
<td><strong>DBMS_XMLQUERY</strong></td>
<td>Create public synonyms for stored procedures</td>
</tr>
<tr>
<td><strong>Microsoft SQL Server</strong></td>
<td>PostgreSQL (Aurora)</td>
<td>Improved accuracy in schema and object definition when a procedure call specifies the database name only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System objects without schemas now convert as is</td>
</tr>
<tr>
<td><strong>Oracle</strong></td>
<td>PostgreSQL (Aurora)</td>
<td>Improved support for global cursors, to allow <code>OPEN</code> and <code>FETCH</code> commands in separate stored routines. With this improvement, the cursor can be opened in one procedure, fetched in another procedure, and closed in a third procedure. Column aliases from a subquery are now converted correctly. Support for the <code>REVERSE</code> function Support for parameters in an anonymous block Added option to convert <code>ROWID</code> as <code>varchar(36)</code>. Using this new option, you can create a <code>DOMAIN</code> for <code>MyRowid</code> as a <code>varchar(36)</code> <code>NOT NULL</code>, then convert the <code>ROWID</code> datatype and column as <code>MyRowid</code>, and finally create a <code>SEQUENCE</code> for each column and default it to <code>NEXTVAL()</code>. System view emulation Support for FORALL MERGE Support global cursor with parameters of <code>%TYPE</code> types Improvements when converting from Oracle: Support global cursor with parameters of <code>%TYPE</code> types Support global cursor with parameters of <code>%TYPE</code> types Support for creation of initializing function for a package without variables <code>AI</code> raised on the routine with 100+ arguments Correct conversion <code>DECODE</code> to <code>CASE</code> with <code>NULL</code> Transform for recursive SQL if a column other than connecting column is needed. Usage of <code>%FOUND</code> and <code>%NOTFOUND</code> attributes of local explicit cursors Calls to functions with the result of a collection type using named notation Functions with the result of a collection type and default argument values</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>
</tbody>
</table>
| Greenplum       | Redshift    | FUNCTION declaration (DROP/CREATE PROCEDURE)  
Variables and constants declaration  
Variable assignments and expressions  
Control-of-Flow Language.FOR  
Cursors  
Dynamic SQL v1. (rejection with AI)  
Datatype mapping Greenplum PL/pgSql- Redshift PL/pgSQL  
PL/pgSQL structure (BEGIN...END, subblocks, labels)  
Comments  
Control-of-Flow Language. RETURN / RETURN NEXT  
Control-of-Flow Language. IF-THEN-ELSE  
Control-of-Flow Language. EXIT / CONTINUE in loops  
Control-of-Flow Language. Simple LOOP  
Control-of-Flow Language. WHILE  
Common Statements. Call another Procedure/UDF  
RETURNS SETOF results in Redshift functions (plpythonu)  
Returns VOID value to Redshift (plythonu) |
| Microsoft SQL Server | Redshift    | Support for CLR stored procedures  
Improved support in stored code for transaction control (COMMIT, ROLLBACK) |
| Netezza         | Redshift    | Netezza procedures no longer convert to python functions  
Support for assignments, SELECT into variable, procedure comments, and other additional cases. |
| Oracle          | Redshift    | Improved support for PL/SQL packages  
Improved support in stored code for transaction control (COMMIT, ROLLBACK) |
<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>What's new, enhanced, or fixed</th>
</tr>
</thead>
</table>
| Teradata | Redshift | Support for transaction control (COMMIT, ROLLBACK) in stored code.  
Execute stored procedures and functions  
Return of result set cursors  
Diagnostic statements  
DDL statement support for CREATE TABLE and CREATE VOLATILE TABLE |

Resolved:
- FIX: Unable to translate SQL Server SYSDATETIMEOFFSET function to PG
- FIX: Service Profile testconnection failing for Amazon ETL Service (SCT 626)
- FIX: Issues converting Teradata table to Redshift
- FIX: Global Service Profile testconnection failing for Amazon ETL Service
- FIX: SCT Assessment Report Can't Save Multi-Byte Characters as PDF
- FIX: DMS standalone agent now supports S3 as target for DMS local task
- FIX: You can now see SAP ASE system tables in the object tree.
- FIX: In conversions from SQL Server to PostgreSQL, SCT now supports comments in try/catch blocks.

Release Notes for the AWS Schema Conversion Tool Build 630

The AWS Schema Conversion Tool 1.0.630 release was merged into the AWS Schema Conversion Tool 1.0.631 release. For changes that were merged together, see Release Notes for the AWS Schema Conversion Tool Build 631 (p. 195).
Release Notes for the AWS Schema Conversion Tool Build 629

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.629.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Redshift**               | For conversions from Netezza, stored code support for the following:  
  • Stored procedures  
  For conversions from Microsoft SQL Server:  
  • Added ability to print source code CLR stored procedure  
  • OBJECT_ID  
  For conversions from Oracle, stored code support for the following:  
  • PL/SQL packages  
  For conversions from Teradata:  
  • Exception handling  
  • Condition handling  
  • DCL statements  
  • DDL statements: CREATE other objects  
  • DML statements: improved support |
| **SAP ASE**                 | Support for SAP ASE 12.5 as a source |
| **DynamoDB**               | For conversions from Cassandra:  
  • Migration improvements and bugfixes  
  • Collection type extraction for tuple and frozen |
| **MySQL 8**                | For conversions from Oracle, added support for hierarchical queries using Oracle CONNECT BY, now converting syntax to WITH Queries (CTE) including RECURSIVE modifier. |
| **MySQL and Aurora MySQL** | For conversions from SAP ASE:  
  • Support for user-defined messages added to the system table sysusermessages for use by any application. Use sp_addmessage to add messages to sysusermessages; use sp_getmessage to retrieve messages for use by print and raiserror.  
  • Corrected wrong conversion of RAISERROR. |
| **PostgreSQL**             | For conversions from Oracle, support for the following: |
### New Feature or Enhancement

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Options to ignore disabled triggers and constraints</td>
</tr>
<tr>
<td>• Assign operation of array or global nested table of nested record</td>
</tr>
<tr>
<td>• EXTRACT, EXTRACTVALUE functions</td>
</tr>
<tr>
<td>• Local and global application contexts with Extension Pack, including additional object category in the object tree and rules.</td>
</tr>
<tr>
<td>• XMLSEQUENCE, XMLELEMENT, and XMLTYPE methods</td>
</tr>
<tr>
<td>• Improved global cursor support to convert OPEN and FETCH commands in separate stored routines. Now, the cursor can be opened in one procedure, fetched in another one, and closed in the third one.</td>
</tr>
<tr>
<td>• Fixed: Columns aliases in subselect are no longer incorrectly converted</td>
</tr>
<tr>
<td>• Fixed SCT Transformer Error complex join no longer shows table on both sides of the join.</td>
</tr>
</tbody>
</table>

### For conversions from Microsoft SQL Server, support for the dynamic SQL

- PostgreSQL and Aurora PostgreSQL
  - For conversions from Oracle, `dbms_application_info.set_action` can be emulated by Extension Pack. Also, where SQL%bulk_exceptions reference caused a transformer error, these exceptions should now be rejected with AI.

### Resolved:

- Triggers and Procedures can now convert from SQL Server to MySQL
- SCT now able to converting a schemas of 6000 procedures in conversions from SQL Server to Aurora PostgreSQL
- Fixed inability to use AWS Glue in N. Virginia region
- SQL Server source procedures no longer get stuck during schema conversion
- General bugfixing and improvements.
Release Notes for the AWS Schema Conversion Tool Build 628

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.628.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
| Service Substitutions     | For conversions from DB2 to Aurora MySQL:  
• Email sending emulation  
• File support emulation  
• Job emulation  

For conversions from DB2 to Aurora PostgreSQL:  
• Email sending emulation  
• File support emulation  
• UTL_File package emulation  

For conversions from Microsoft SQL Server to MySQL, PostgreSQL, Aurora MySQL, or Aurora PostgreSQL:  
• SQL Server Scheduler emulation  

For conversions from Oracle to MySQL or Aurora MySQL:  
• File support emulation  
• OWA_ packages emulation  
• Oracle queuing emulation  
• HTP/HTF/OWA_* packages emulation  

For conversions from Oracle to PostgreSQL or Aurora PostgreSQL:  
• File support emulation  
• OWA_ packages emulation  
• Oracle queuing emulation  
• UTL_FILE package emulation  

Redshift  
For conversions from Microsoft SQL Server, stored code support for the following:  
• Stored procedures and dynamic SQL  

For conversions from Oracle, stored code support for the following:  
• Cursors in procedures  
• MERGE
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
|                           | • EXECUTE IMMEDIATE is supported in procedures and has improved support in dynamic SQL  
|                           | • Nested subprograms in procedures |
| For conversions from Teradata, stored code support for the following: |
|                           | • Cursors in procedures  
|                           | • MERGE  
|                           | • QUALIFY  
|                           | • DECLARE and SET variables |
| SAP ASE 15.0              | Support for SAP ASE 15.0 as a source |
| DynamoDB                  | For conversions from Cassandra, support for the following: |
|                           | • Collection types (SET, LIST, MAP) extraction  
<p>|                           | • General improvements and fixes |
| MySQL 8                   | For conversions from DB2 or SAP ASE, support for case-sensitive instance |</p>
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
| PostgreSQL 11            | **For conversions from DB2, support for the following:**  
  • Support for embedded transactions and `CALL` for SQL stored procedures. In procedures invoked by the `CALL` command as well as in anonymous code blocks (DO command), it is possible to end transactions using the commands `COMMIT` and `ROLLBACK`. A new transaction is started automatically after a transaction is ended using these commands, so there is no separate `START TRANSACTION` command. (Note that `BEGIN` and `END` have different meanings in PL/pgSQL.)  

**For conversions from Microsoft SQL Server, support for the following:**  
• Support for dynamic T-SQL. Support for calling stored procedures by using `CALL` in dynamic queries.  

**For conversions from Oracle, support for the following:**  
• Support for SQL*Plus conversions, including procedures and dynamic SQL. Stored Procedures `CALL` in dynamic queries.  
• Added option to change default data type mapping for columns with PK and FK referring to a modified PK. For example, you can convert a number with scale and precision to a bigint, for performance improvement. Previously, when the number data type was migrated from Oracle to PostgreSQL, SCT defaulted the conversion to numeric in PostgreSQL. If the number in Oracle was being used with scale and precision, this conversion was okay. But, if it was an integer with a PK index on the column, converting it to numeric in PostgreSQL could cause performance issues.  
• Support for converting `%TYPE` to built-in datatype. SCT replaces `%TYPE` with new datatype, according to type mapping rules.  
• Replaced default mapping of function and procedure arguments so that `NUMBER` converts to `NUMERIC`, rather than to `DOUBLE PRECISION`  

**For conversions from SAP ASE, support for the following:**  
• `CREATE PROCEDURE` with `RETURN`  
• Support for calling stored procedures by using `CALL` in dynamic queries  
• Support for `FOREIGN KEY` on partitioned tables. PostgreSQL 11 only supports foreign keys from a partitioned table to a (non-partitioned) table.  
• Support for indexes on partitioned tables. After you index the master table, PostgreSQL automatically creates an identically configured index on existing child partitions and on future partitioned tables.  
• Support for `PRIMARY KEY` on partitioned tables. In version 11, you can add a primary key to the master table. Doing this creates the `PRIMARY KEY` on all existing child tables and future partition
New Feature or Enhancement | Description
--- | ---
 | tables. If PARTITION is added, there's no need to create a primary key manually.
 | • Support for triggers on partitioned tables. After you create a trigger on the master table, PostgreSQL automatically creates the trigger on all child tables (this behavior is similar to the one for indexes).

| PostgreSQL and Aurora PostgreSQL | For conversions from Sybase, UDT is converted to DOMAIN

Resolved:

* Added support for creating an SCT report along with logical components in WQF.
* Fix: Issue where WQF report included extraction failure action items
* Fix: Issue when SCT not automatically picking up DMS endpoint for Aurora PostgreSQL
* Fix: Issue when **Apply to database** is grayed out for multiple schemas
* General bugfixing and improvements
Release Notes for the AWS Schema Conversion Tool Build 627

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.627.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
| Support for conversions to stored procedures in Amazon Redshift | Improving support for the following features when converting stored procedures from Microsoft SQL Server to Amazon Redshift:  
  - Improvements to DDL support  
  - Temporary tables  
  - PROCEDURE declaration  
  - Procedure error handling  
  - SET statements  
  - Control-of-flow language  
  - Operators  
  - CURSORS  
  - Arithmetic expressions with mixed types of operands |
| PostgreSQL 11 | Improvements for converting to PostgreSQL include the following.  
  - For DB2 sources:  
    - DEFAULT partition feature stores tuples that don’t map to any other partition. Prior to PostgreSQL 11, these rows would error out. A row that is not mapped to any partition table would be inserted in the default partition.  
    - Covering indexes can now be created, using the INCLUDE clause of CREATE INDEX  
    - CREATE PROCEDURE defines a new procedure. CREATE OR REPLACE PROCEDURE will either create a new procedure, or replace an existing definition. To be able to define a procedure, the user must have the USAGE privilege on the language.  
    - Once the trigger is created on the master table, it will automatically create the trigger on all child tables (this behavior is similar to the one seen for index).  
    - Once the index is created on the master table, it will automatically create the index with the same configuration on all existing child partition and take care of any future partition tables as well.  
    - Postgres 11 only supports foreign keys from a partitioned table to a (non-partitioned) table  
    - In version 11 PRIMARY KEY can be added to the master table which will create the PRIMARY KEY on all existing child tables and future partition tables. IF PARTITION is added, no need to create primary keys manually  
  - For Microsoft SQL Server sources:  
    - Covering indexes can now be created, using the INCLUDE clause of CREATE INDEX |
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>• In procedures invoked by the <strong>CALL</strong> command as well as in anonymous code blocks (<strong>DO</strong> command), it is possible to end transactions using the commands <strong>COMMIT</strong> and <strong>ROLLBACK</strong>. A new transaction is started automatically after a transaction is ended using these commands, so there is no separate <strong>START TRANSACTION</strong> command. (Note that <strong>BEGIN</strong> and <strong>END</strong> have different meanings in PL/pgSQL.)</td>
<td></td>
</tr>
<tr>
<td>• <strong>CALL</strong> syntax for SQL stored procedures</td>
<td></td>
</tr>
<tr>
<td>• <strong>CREATE Procedure</strong> defines a new procedure. <strong>CREATE OR REPLACE Procedure</strong> will either create a new procedure, or replace an existing definition. To be able to define a procedure, the user must have the <strong>USAGE</strong> privilege on the language.</td>
<td></td>
</tr>
<tr>
<td>• Once the trigger is created on the master table, it will automatically create the trigger on all child tables (this behavior is similar to the one seen for index).</td>
<td></td>
</tr>
<tr>
<td>• Once the index is created on the master table, it will automatically create the index with the same configuration on all existing child partition and take care of any future partition tables as well.</td>
<td></td>
</tr>
<tr>
<td>• Postgres 11 only supports foreign keys from a partitioned table to a (non-partitioned) table</td>
<td></td>
</tr>
<tr>
<td>• In version 11 <strong>PRIMARY KEY</strong> can be added to the master table which will create the <strong>PRIMARY KEY</strong> on all existing child tables and future partition tables. If <strong>PARTITION</strong> is added, no need to create primary keys manually</td>
<td></td>
</tr>
<tr>
<td>• <strong>For Oracle sources:</strong></td>
<td></td>
</tr>
<tr>
<td>• In procedures invoked by the <strong>CALL</strong> command as well as in anonymous code blocks (<strong>DO</strong> command), it is possible to end transactions using the commands <strong>COMMIT</strong> and <strong>ROLLBACK</strong>. A new transaction is started automatically after a transaction is ended using these commands, so there is no separate <strong>START TRANSACTION</strong> command. (Note that <strong>BEGIN</strong> and <strong>END</strong> have different meanings in PL/pgSQL.)</td>
<td></td>
</tr>
<tr>
<td>• <strong>CALL</strong> syntax for SQL stored procedures</td>
<td></td>
</tr>
<tr>
<td>• <strong>CREATE Procedure</strong> defines a new procedure. <strong>CREATE OR REPLACE Procedure</strong> will either create a new procedure, or replace an existing definition. To be able to define a procedure, the user must have the <strong>USAGE</strong> privilege on the language.</td>
<td></td>
</tr>
<tr>
<td>• <strong>For SAP ASE sources:</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>CALL Syntax for SQL Stored Procedures</strong></td>
<td></td>
</tr>
<tr>
<td>• In procedures invoked by the <strong>CALL</strong> command as well as in anonymous code blocks (<strong>DO</strong> command), it is possible to end transactions using the commands <strong>COMMIT</strong> and <strong>ROLLBACK</strong>. A new transaction is started automatically after a transaction is ended using these commands, so there is no separate <strong>START TRANSACTION</strong> command. (Note that <strong>BEGIN</strong> and <strong>END</strong> have different meanings in PL/pgSQL.)</td>
<td></td>
</tr>
<tr>
<td>• <strong>CREATE Procedure</strong> defines a new procedure. <strong>CREATE OR REPLACE Procedure</strong> will either create a new procedure, or replace an existing definition. To be able to define a procedure, the user must have the <strong>USAGE</strong> privilege on the language.</td>
<td></td>
</tr>
</tbody>
</table>

Version 1.0

206
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
|                           | • **UPDATE** statements that change a partition key column now cause affected rows to be moved to the appropriate partitions  
• The PostgreSQL 11 **DEFAULT** partition feature stores tuples that don't map to any other partition. Prior to PostgreSQL 11, these rows would error out. A row that is not mapped to any partition table would be inserted in the default partition.  
• Once the index is created on the master table, it will automatically create the index with the same configuration on all existing child partition and take care of any future partition tables as well.  
• Hash Partitioning – The table is partitioned by specifying a modulus and a remainder for each partition. Each partition will hold the rows for which the hash value of the partition key divided by the specified modulus will produce the specified remainder |
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL 8</td>
<td>SCT now supports case sensitive instances for conversions from Oracle, Microsoft SQL Server, Azure, PostgreSQL, and MySQL. SCT now offers the option to create and use database, table, and trigger names in lower case.</td>
</tr>
</tbody>
</table>

Improvements for converting to MySQL include the following.

- **For DB2 sources:**
  - MySQL now supports descending indexes: `DESC` in an index definition is no longer ignored but causes storage of key values in descending order. Previously, indexes could be scanned in reverse order but at a performance penalty. A descending index can be scanned in forward order, which is more efficient. Descending indexes also make it possible for the optimizer to use multiple-column indexes when the most efficient scan order mixes ascending order for some columns and descending order for others.
  - MySQL now supports creation of functional index key parts that index expression values rather than column values. Functional key parts enable indexing of values that cannot be indexed otherwise, such as JSON values.
  - MySQL supports now `CTE` and recursive `CTE WITH (Common Table Expressions)`.
  - A derived table now may be preceded by the `LATERAL` keyword to specify that it is permitted to refer to (depend on) columns of preceding tables in the same `FROM` clause. Lateral derived tables make possible certain SQL operations that cannot be done with nonlateral derived tables or that require less-efficient workarounds.
  - Several existing aggregate functions now can be used as window functions.
    - `AVG()`
    - `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.
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>• CUME_DIST()</td>
<td></td>
</tr>
<tr>
<td>• DENSE_RANK()</td>
<td></td>
</tr>
<tr>
<td>• FIRST_VALUE()</td>
<td></td>
</tr>
<tr>
<td>• LAG()</td>
<td></td>
</tr>
<tr>
<td>• LAST_VALUE()</td>
<td></td>
</tr>
<tr>
<td>• LEAD()</td>
<td></td>
</tr>
<tr>
<td>• NTH_VALUE()</td>
<td></td>
</tr>
<tr>
<td>• NTILE()</td>
<td></td>
</tr>
<tr>
<td>• PERCENT_RANK()</td>
<td></td>
</tr>
<tr>
<td>• RANK()</td>
<td></td>
</tr>
<tr>
<td>• ROW_NUMBER()</td>
<td></td>
</tr>
</tbody>
</table>

- **For Microsoft SQL Server and Azure SQL DB sources:**
  - Several existing aggregate functions now can be used as window functions.
    - AVG()
    - 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()

- **WITH (Common Table Expressions)** – MySQL supports CTE and Recursive CTE
  - MySQL supports window functions that, for each row from a query, perform a calculation using rows related to that row.
    - CUME_DIST()
    - FIRST_VALUE()
    - LAG()
    - LAST_VALUE()
    - LEAD()
  - Lateral derived tables
  - MySQL now supports descending indexes: DESC in an index definition is no longer ignored but causes storage of key values in descending order. Previously, indexes could be scanned in
New Feature or Enhancement | Description
---|---
reverse order but at a performance penalty. A descending index can be scanned in forward order, which is more efficient. Descending indexes also make it possible for the optimizer to use multiple-column indexes when the most efficient scan order mixes ascending order for some columns and descending order for others
MySQL now supports use of expressions as default values in data type specifications. This includes the use of expressions as default values for the BLOB, TEXT, GEOMETRY, and JSON data types, which previously could not be assigned default values at all.
For Oracle sources:
A derived table now may be preceded by the LATERAL keyword to specify that it is permitted to refer to (depend on) columns of preceding tables in the same FROM clause. Lateral derived tables make possible certain SQL operations that cannot be done with nonlateral derived tables or that require less-efficient workarounds
InnoDB supports NOWAIT and SKIP LOCKED options with SELECT ... FOR SHARE and SELECT ... FOR UPDATE locking read statements. NOWAIT causes the statement to return immediately if a requested row is locked by another transaction. SKIP LOCKED removes locked rows from the result set.
WITH (Common Table Expressions) – MySQL supports CTE and recursive CTE
Several existing aggregate functions now can be used as window functions.
• AVG()
• 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 implements regular expression support as follows.
• NOT REGEXP
• REGEXP
• REGEXP_INSTR()
• REGEXP_LIKE()
• REGEXP_REPLACE()
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>• REGEXP_SUBSTR()</td>
<td></td>
</tr>
<tr>
<td>• RLIKE</td>
<td></td>
</tr>
<tr>
<td>• MySQL supports window functions that, for each row from a query, perform a calculation using rows related to that row.</td>
<td></td>
</tr>
<tr>
<td>• CUME_DIST()</td>
<td></td>
</tr>
<tr>
<td>• DENSE_RANK()</td>
<td></td>
</tr>
<tr>
<td>• FIRST_VALUE()</td>
<td></td>
</tr>
<tr>
<td>• LAG()</td>
<td></td>
</tr>
<tr>
<td>• LAST_VALUE()</td>
<td></td>
</tr>
<tr>
<td>• LEAD()</td>
<td></td>
</tr>
<tr>
<td>• NTH_VALUE()</td>
<td></td>
</tr>
<tr>
<td>• NTILE()</td>
<td></td>
</tr>
<tr>
<td>• PERCENT_RANK()</td>
<td></td>
</tr>
<tr>
<td>• RANK()</td>
<td></td>
</tr>
<tr>
<td>• ROW_NUMBER()</td>
<td></td>
</tr>
<tr>
<td>• For SAP ASE sources:</td>
<td></td>
</tr>
<tr>
<td>• MySQL now supports descending indexes: DESC in an index definition is no longer ignored but causes storage of key values in descending order. Previously, indexes could be scanned in reverse order but at a performance penalty. A descending index can be scanned in forward order, which is more efficient. Descending indexes also make it possible for the optimizer to use multiple-column indexes when the most efficient scan order mixes ascending order for some columns and descending order for others.</td>
<td></td>
</tr>
<tr>
<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 that cannot be indexed otherwise, such as JSON values.</td>
<td></td>
</tr>
<tr>
<td>• MySQL now supports use of expressions as default values in data type specifications. This includes the use of expressions as default values for the BLOB, TEXT, GEOMETRY, and JSON data types, which previously could not be assigned default values at all.</td>
<td></td>
</tr>
</tbody>
</table>

AWS Glue

Improvements for converting to AWS Glue include the following.

• Improved conversion of procedures parameters.
• Ability to add quotes for special characters to the AWS Glue scripts.
• Fixes, improvements and polishing for conversion from Teradata BTEQ to AWS Glue.
• Improvements and fixes for conversion from Oracle to AWS Glue.

DynamoDB

Migration improvements and fixes for migrations from Cassandra to DynamoDB.

Oracle

SCT now supports INTERVAL partitioning in Oracle sources.

Resolved:

• Added support for MySQL 8 as a destination in AWS SCT.
• Added support for Windows Authentication in WQF.
• Username/Password fields are no longer required for AWS DMS proxy.
• Fix: SCT now able to read all the database objects when entire schema is selected (Oracle to Amazon Aurora PostgreSQL).
• Fix: Physical path to AWS SCT statistics file now supported in WQF, when you are loading physical/logical components from CSV/JSON files.
• Fix: Composite key (b,a) no longer changes to (a,b) while converting from SQL Server to Aurora MySQL.
• Fix: View conversion error (SQL Server to PostgreSQL).
• Fix: For conversions from SAP ASE, added ability to specify the character set on the server connection.
Release Notes for the AWS Schema Conversion Tool Build 626

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.626.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
| PostgreSQL 11             | PostgreSQL 11 is now supported as a target. Key features include the following:  
  • Creating a "default" partition for rows that don't automatically fit into an existing partition.  
  • Creating foreign keys from a partitioned table to a non-partitioned table.  
  • Creating a unique index on the master table. This index manages indexes with the same configuration on existing child tables and future partitioned tables.  
  • When you create a trigger on the master table, it automatically creates triggers on all child tables.  
  • When you UPDATE a partition key column so that it now fits on a different partition, that row is moved to the appropriate partition.  
  • Using HASH partitions by specifying a modulus to apply to the partition key. The remainder of the modulus is used to identify the partition for that row. |
| MySQL 8.0                 | MySQL 8.0 is now supported as a target. Key features include the following:  
  • Using DESC in an index definition now stores key values in descending order. Previously, scanning these indexes in reverse order caused a performance penalty. Now, a descending index can be scanned efficiently (moving forward, rather than in reverse). Descending indexes also make it possible for the optimizer to use multiple-column indexes when the most efficient scan order mixes ascending order for some columns and descending order for others.  
  • Using expressions as default values in data type specifications. This includes BLOB, TEXT, GEOMETRY, and JSON data types.  
  • Creating functional index key parts that index expression values rather than column values. Functional key parts enable indexing of values that can't be indexed otherwise, such as JSON values. |
| SAP ASE 15.5              | SAP ASE 15.5 is now supported as a source. |
| AzureSQL/MSSQL to MySQL/AuroraMySQL | You can now use identity columns in temporary tables and table-valued functions.  
You can now use INSERT INTO for tables with an identity column. |
| Cassandra to DynamoDB     | Migration improvements and bugfixes.  
Adding compaction process in the wizard. |
New Feature or Enhancement | Description
---|---
Replaced utility telnet with netcat in the wizard.
DB2 to MariaDB 10.3 | (SQL MODE=ORACLE) Module routines and variables references now supported.
DB2 to MySQL/AuroraMySQL | Service substitution - support for files.
Netezza to Redshift | Table columns that contains DEFAULT 'now(0)' are now set to DEFAULT SYSDATE.
Oracle to MariaDB 10.3 | (SQL MODE=ORACLE) Module routines, variable references, and using sequences now supported.
Oracle to MySQL/AuroraMySQL | Service substitution - support for files.
Oracle to AWS Glue | The built-in function for FORMAT MODELS set is now supported.
SAP ASE to MySQL/Aurora MySQL | Global variables are now supported.
SAP ASE to PostgreSQL/Aurora PostgreSQL | Global variables are now supported.
Teradata BTEQ to AWS Glue | Now supporting built-in functions for BITBYTE and PERIOD sets.

Issues Resolved:
- Fixed internal converter errors for migrating SQL Server to MySQL.
- Fixed issues in the Application Conversion Assessment report.
- Fixed issues for Oracle interval partitioning.
- Fixed and polished conversions to AWS Glue.
- General bugfixing and improvements.
Release Notes for the AWS Schema Conversion Tool Build 625

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.625.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata BTEQ conversion</td>
<td>You can now convert Teradata BTEQ scripts to AWS Glue.</td>
</tr>
<tr>
<td>DB2 to MariaDB 10.3</td>
<td>You can now use cursors with parameters.</td>
</tr>
</tbody>
</table>
| DB2 to MariaDB 10.3 (SQL MODE=ORACLE) | For conversions from DB2 to MariaDB 10.3 using SQL MODE=ORACLE, the following are now supported:  
  * %TYPE and %ROWTYPE variables  
  * ROW structure variables in a local cursors  
  * Declaring scalar variables  
  * Declaring a default variable value  
  * Converting modules to packages.  
  * CREATE PROCEDURE p1 (param OUT INT)  
  * CREATE PROCEDURE p1 (a IN OUT INT)  
  * Using AS before function body  
  * Using IS before function body. |
| Oracle to MariaDB 10.3     | For conversions from Oracle to MariaDB 10.3, you can use %TYPE and %ROWTYPE variables. |
| Oracle to MariaDB 10.3 (SQL MODE=ORACLE) | For conversions from Oracle to MariaDB 10.3 using SQL MODE=ORACLE, the following are now supported:  
  * %TYPE and %ROWTYPE variables  
  * Package routines (procedures and functions)  
  * SELECT UNIQUE  
  * GLOBAL CURSORS  
  * Package Initialization block BEGIN..END  
  * Package without the body  
  * Local variables and constants  
  * Global variables and constants |
| MS SQL Server to MySQL/AuroraMySQL | For conversions from MS SQL Server to MySQL/AuroraMySQL, the following are now supported:  
  * sp_prepexec() with dynamic SQL  
  * sp_execute() with dynamic SQL  
  * sp_unprepare() with dynamic SQL |
| SAP ASE 15.7               | SAP ASE 15.7 is now supported. |

Issues Resolved:
- Added notifications for when you close a project and might lose data agent connections.
- Resolved conversion issues for Oracle to PostgreSQL.
- Resolved issues with SCT Oracle to PostgreSQL - to_date issue.
- General bug fixes and improvements.
Release Notes for the AWS Schema Conversion Tool Build 624

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.624.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert Oracle ETL to AWS Glue</td>
<td>You can now convert Oracle ETL jobs to ETL based on AWS Glue to work in conjunction with Amazon Redshift.</td>
</tr>
<tr>
<td>Microsoft SQL Server, Oracle, and IBM Db2 LUW to Amazon RDS for MariaDB conversions</td>
<td>Added support for RDS for MariaDB 10.2 and 10.3 as a target.</td>
</tr>
<tr>
<td>SAP ASE to RDS for MySQL and Amazon Aurora with MySQL compatibility conversions</td>
<td>Added support for SAP ASE to MySQL database object conversions.</td>
</tr>
<tr>
<td>Support for Oracle extension during Oracle conversion to PostgreSQL</td>
<td>You can now use the Oracle extension as a destination when converting to PostgreSQL in Oracle compatibility mode.</td>
</tr>
</tbody>
</table>

Issues Resolved:

- Fixed an issue while using AWS profiles in AWS SCT in the Beijing and Ningxia regions.
- Fixed an issue in SQL Server to RDS for PostgreSQL and Aurora PostgreSQL conversions where SCT doesn't return when converting a stored procedure.
- Fixed a virtual partition error in data extractor agents from build 623.
- General bug fixes and improvements.
Release Notes for the AWS Schema Conversion Tool Build 623

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.623.

**Note**
In the following table, interpret "PostgreSQL" to mean both PostgreSQL and Amazon Aurora PostgreSQL.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP ASE conversion to PostgreSQL</td>
<td>Added SAP ASE as a source database.</td>
</tr>
<tr>
<td>Oracle function NLSSORT emulation for conversion to PostgreSQL</td>
<td>The NLSSORT function is used to replace a character string with the equivalent sort string used by the linguistic sort mechanism. In the case of a binary sort; the sort string is the same as the input string. In linguistic sort; each character in a string is replaced by some other binary values.</td>
</tr>
<tr>
<td>Oracle function DBTIMEZONE emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL DBTIMEZONE function returns the database time zone as a time zone offset (in the following format: <code>[+-]TZH:TZM</code>) or a time zone region name.</td>
</tr>
<tr>
<td>Oracle function LNNVL emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL LNNVL function is used in the WHERE clause of a SQL statement to evaluate a condition when one of the operands may contain a NULL value.</td>
</tr>
<tr>
<td>Oracle function TO_SINGLE_BYTE emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL TO_SINGLE_BYTE function returns a character value with all of the multibyte characters converted to single-byte characters. To use this function; your database character set contains both single-byte and multibyte characters.</td>
</tr>
<tr>
<td>Oracle function TO_MULTI_BYTE emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL TO_MULTI_BYTE function returns a character value with all of the single-byte characters converted to multibyte characters. To use this function; your database character set contains both single-byte and multibyte characters.</td>
</tr>
<tr>
<td>Oracle function ROUND(date [, fmt ]) emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL ROUND function returns a date rounded to a specific unit of measure.</td>
</tr>
<tr>
<td>Oracle function MEDIAN emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL MEDIAN function returns the median of an expression.</td>
</tr>
<tr>
<td>Oracle function LISTAGG emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL LISTAGG function concatenates values of the measure_column for each GROUP based on the order_by_clause.</td>
</tr>
<tr>
<td>Oracle function NANVL emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL NANVL function lets you substitute a value for a floating point number such as BINARY_FLOAT or BINARY_DOUBLE when a NaN (&quot;Not a Number&quot;) value is encountered. This is most commonly used to convert NaN values into either NULL or 0.</td>
</tr>
<tr>
<td>New Feature or Enhancement</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Oracle function INSTRB emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL INSTRB function returns the location of a substring in a string, using bytes instead of characters.</td>
</tr>
<tr>
<td>Oracle function SUBSTRB emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL SUBSTRB function return a portion of string; beginning at a specified position in the string. The function SUBSTRB calculates lengths using bytes to return.</td>
</tr>
<tr>
<td>Oracle function LENGTHB emulation for conversion to PostgreSQL</td>
<td>The Oracle/PLSQL LENGTHB function returns the length of the specified string, using bytes instead of characters.</td>
</tr>
<tr>
<td>MSSQL: Convert scripts to PostgreSQL</td>
<td>Convert T-SQL scripts, DML, and DDL to equivalent code or components.</td>
</tr>
<tr>
<td>MSSQL: OBJECT_NAME(@@PROCID) emulation for conversion to PostgreSQL</td>
<td>The MS SQL Server OBJECT_NAME(@@PROCID) function returns the object identifier (ID) of the current T-SQL module.</td>
</tr>
<tr>
<td>MSSQL: OBJECT_SCHEMA_NAME (@@PROCID) emulation for conversion to PostgreSQL</td>
<td>The Microsoft SQL Server OBJECT_SCHEMA_NAME(@@PROCID) function returns the database schema name for schema-scoped objects.</td>
</tr>
<tr>
<td>MSSQL: Sysobject view emulation for conversion to MySQL, Aurora MySQL, or MariaDB</td>
<td>The sysobject view can be emulated by Ext.Pack during a conversion from MS SQL Server to MySQL, Aurora MySQL, or MariaDB.</td>
</tr>
<tr>
<td>Modified SCT &quot;Compare mode&quot; for DDL Replication Module for PostgreSQL</td>
<td>When you're comparing two PostgreSQL schemas with SCT, you no longer need to deploy the service schema.</td>
</tr>
<tr>
<td>Improved the &quot;small tables&quot; rule (DISTSTYLE=ALL) for Redshift</td>
<td>The small tables rule takes into account the number of rows in the table, rather than its size.</td>
</tr>
<tr>
<td>Improved ZSTD compression control</td>
<td>Depending on the settings, the SCT sets the default ZSTD compression in all columns that are not included in the SortKey (DistKey - optionally).</td>
</tr>
</tbody>
</table>

Issues Resolved:

- Fixed an issue with double UTF-8 encoding in the data extractors for migrations from Netezza to Redshift.
- Fixed an issue with unsafe YAML deserialization in Cassandra configuration files.
- Improvements to converting from Microsoft SQL Server to PostgreSQL.
- Fixed an issue where the connection to Cassandra cluster fails if the version is 2.1.20. SCT now supports migrating from Cassandra 2.1.20.
# Release Notes for the AWS Schema Conversion Tool Build 622

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.622.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
| AWS DMS agent for AWS SCT | If you're using the local AWS DMS agent to migrate data to an AWS Snowball Edge device or to Amazon S3, note the following:  
  - If you have existing tasks running with local DMS agent version 2.4.1, you should migrate these tasks to the latest version of the DMS agent.  
  - Don't upgrade the AWS Schema Conversion Tool until after the DMS agent is upgraded.  
  - The following components must use the same replication engine version number:  
    - The local DMS agent.  
    - The DMS instance for the remote task. |
| Microsoft SQL Server to PostgreSQL: Script conversion | When you use SQL Server as a source, a new item appears in the source pane: "SQL Scripts'. If you choose Actions | Load Scripts from the AWS SCT menu, you can choose one or more .sql files to convert. During conversion, AWS SCT rewrites the scripts for use with PostgreSQL. The results are shown in the target pane: "SQL Scripts". The AWS SCT assessment report highlights any issues encountered when it converts the scripts. |
| Data extractors allow NULL value replacement | The COPY command can replace NULLs with a different value:  
  - COPY ... FROM ... NULL AS 'MyReplacementValue' ...  
  - CREATE EXTERNAL TABLE ... USING (REMOTESOURCE ... NULLVALUE 'MyReplacementValue' ...) |
| Support for ZSTD compression in Redshift | Ability to choose ZSTD compression for table columns. New project settings for choosing conversion options:  
  - Do not use ZSTD compression.  
  - Use ZSTD compression for all columns.  
  - Use ZSTD compression for all columns except Distkey and Sortkey. |
| Oracle data warehouse: Support for multiple files during script generation | New project settings for saving SQL scripts:  
  - Single file - all SQL will be generated to a single file.  
  - Multiple files - each file contains SQL statements, including ID and stage name. The scripts should be run one at a time, in ID order. |
| Workload Qualification Framework | The Workload Qualification Framework (WQF) is part of AWS Database Migration Service (AWS DMS). It helps you analyze and plan your migrations to AWS database services. For more information,
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>see Creating Migration Reports with the Workload Qualification Framework (p. 81).</td>
</tr>
</tbody>
</table>

**Issues Resolved**

- Microsoft SQL Server: Unable to create assessment reports for a named instance.
- Oracle to PostgreSQL: Error message incorrectly states that materialized views aren't supported.
- Microsoft SQL Server to PostgreSQL: Dynamic SQL being converted incorrectly.
- Assessment reports: PDF files containing multibyte characters can't be saved.
- Microsoft SQL Server to PostgreSQL: Conversions fail with NullPointerException.
- Using AWS SCT to create a task in AWS DMS: Bug fixes. We recommend upgrading to AWS SCT build 622 if you need to run an AWS DMS task.
Release Notes for the AWS Schema Conversion Tool Build 621

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.621.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft SQL Server to MySQL: Enhancements</td>
<td>The following date and time functions are supported by the AWS SCT Extension Pack: <code>DATETIMEFROMPARTS</code>, <code>DATETIME2FROMPARTS</code> and <code>TIMEFROMPARTS</code>.&lt;br&gt;The <code>CHECKSUM</code> function is now supported.</td>
</tr>
<tr>
<td>Microsoft SQL Server to PostgreSQL: Enhancements</td>
<td>The <code>OBJECT_ID</code> function is supported by the AWS SCT Extension Pack.&lt;br&gt;In SQL Server, you can specify <code>ELEMENTS XSNIL</code> in a query to generate XML elements, even for NULL values in the result set. AWS SCT recognizes the <code>XSNIL</code> directive when converting from SQL Server to PostgreSQL, and converts it according to PostgreSQL semantics.&lt;br&gt;AWS SCT supports the use of the <code>NEWID</code> function in saved SQL scripts. To enable this behavior:&lt;br&gt;• From the AWS SCT menu bar, choose **Settings</td>
</tr>
<tr>
<td>Saving SQL for source tree creation</td>
<td>You can save the SQL statements that AWS SCT generates to create the source tree. In this release, the following database engines are now supported: MySQL, PostgreSQL, and Azure.</td>
</tr>
<tr>
<td>Support for Amazon Aurora PostgreSQL 10 as a target</td>
<td>Amazon Aurora with PostgreSQL Compatibility now supports PostgreSQL major version 10.4. AWS SCT fully supports using this new release as a target.</td>
</tr>
<tr>
<td>Netezza as a source: Enhancements</td>
<td>AWS SCT recognizes <code>EXTERNAL</code> tables in Netezza. This allows AWS SCT to access external data, as if it were stored natively in Netezza.&lt;br&gt;For faster performance, AWS SCT uses parallel compression when reading data from Netezza.&lt;br&gt;The AWS SCT extraction agent uses the Redshift <code>COPY</code> command to copy data from Netezza to Redshift. You can control <code>COPY</code> command behavior using the following parameters:&lt;br&gt;• <code>extractor.redshift.copying.thread.pool.size</code>—the number of <code>COPY</code> commands that the agent can run simultaneously. The default is 1, meaning that the agent will run all <code>COPY</code> commands sequentially.&lt;br&gt;• <code>extractor.redshift.copy.command.count.threshold</code>—the maximum number of <code>COPY</code> commands that can run at the same time. The default value is 5.</td>
</tr>
<tr>
<td>New Feature or Enhancement</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>When the extraction agent reads data from a single large table, it writes the data to multiple, smaller tables in Amazon Redshift. At the end of the migration, these staging tables are consolidated into a single large table in Redshift. To limit the number of staging tables allowed in Redshift, you can modify the extractor.redshift.staging.table.count.threshold parameter. The default value is 100. (For more information on how to set these parameters, see Configuring Extraction Agents (p. 140).)</td>
</tr>
</tbody>
</table>

Issues Resolved:

- Microsoft SQL Server to MySQL: AWS SCT supports arithmetic operations with mixed types of operands.
- AWS SCT virtual partitioning fully supports columns that contain NULL data. For more information, see Using Virtual Partitioning with AWS Schema Conversion Tool (p. 154).
The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.620.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
</table>
| Save CREATE statements in SQL scripts | AWS SCT allows you to save SQL statements that are used for creating database objects (CREATE TABLE, CREATE VIEW, CREATE FUNCTION, and so on). You can save all of the statements in one SQL script, or save each statement in a separate SQL script. single file, or save each statement in its own SQL script.  

To modify this behavior, choose Settings | Project settings from the AWS SCT menu bar, and choose Save as SQL and Apply. |
| Microsoft SQL Server to MySQL: Support for CHECKSUM, ISNUMERIC and NEWSEQUENTIALID functions | These functions are supported by the AWS SCT Extension Pack. |
| Oracle to MySQL: MERGE Support | The Oracle MERGE statement is used to insert, update or delete rows, depending on whether those same rows currently exist in the table. (This operation is sometimes referred to as an “upsert”.) MySQL doesn't support the MERGE statement; however, AWS SCT can migrate MERGE statements to MySQL-compatible INSERT, UPDATE and DELETE statements. |
| Microsoft SQL Server to MySQL: TIMESTAMP data type | AWS SCT can migrate TIMESTAMP data type from SQL Server to MySQL. |
| Microsoft SQL Server to PostgreSQL: OPENXML functions | AWS SCT can migrate OPENXML functions from SQL Server to MySQL. |
| Oracle to MySQL: Global cursors | Oracle supports global cursors, where a cursor defined in one program unit (stored procedure, function or trigger) can be accessed outside of the program unit where it is defined. MySQL doesn't support global cursors; however, AWS SCT provides equivalent functionality by modifying the program unit code on the MySQL target. |
| Oracle to MySQL: Spatial data | AWS SCT can migrate spatial data, objects and functions from Oracle to MySQL. |
| Oracle to MySQL: TIMESTAMP WITH TIME ZONE | This data type is supported by the AWS SCT Extension Pack. |
| User-defined LOB size with AWS Snowball | You can now define the maximum size for large objects (LOBs) when using the AWS SCT DMS Agent and AWS Snowball. |
Issues Resolved

- Oracle to PostgreSQL: Minor conversion issues fixed.
- partitions no longer drop nulls.
Release Notes for the AWS Schema Conversion Tool Build 619

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.619.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migration support from Apache Cassandra to Amazon DynamoDB</td>
<td>You can use AWS SCT with an AWS data extraction agent to migrate from Apache Cassandra to Amazon DynamoDB. For more information, see Migrating Data From Apache Cassandra to Amazon DynamoDB (p. 159).</td>
</tr>
<tr>
<td>Support for Vertica 9 as a migration source</td>
<td>You can now use Vertica version 9 as a migration source. For more information, see Using Vertica as a Source for AWS Schema Conversion Tool (p. 72).</td>
</tr>
<tr>
<td>Microsoft SQL Server to MySQL: Stored procedure RETURN value support</td>
<td>MySQL doesn't support the RETURN keyword in stored procedures; however, OUT parameters are supported.</td>
</tr>
<tr>
<td>Support for Azure SQL Database as a source</td>
<td>You can now migrate from Azure SQL Database to MySQL, PostgreSQL, or Amazon Aurora (with MySQL or PostgreSQL compatibility).</td>
</tr>
</tbody>
</table>

Issues Resolved

- Microsoft SQL Server to PostgreSQL: AWS SCT supports arithmetic operations with mixed types of operands.
- Oracle Advanced Queuing (AQ) is supported when migrating to Amazon RDS for Oracle.
## Release Notes for the AWS Schema Conversion Tool Build 618

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.618.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic SQL in string literals</td>
<td>You can execute a command string or character string within a Transact-SQL batch, or one of the following modules: system stored procedure, user-defined stored procedure, CLR stored procedure, scalar-valued user-defined function, or extended stored procedure. The EXECUTE statement can be used to send pass-through commands to linked servers. Additionally, the context in which a string or command is executed can be explicitly set. Metadata for the result set can be defined by using the WITH RESULT SETS options.</td>
</tr>
<tr>
<td>MS SQL Server to PostgreSQL: Spatial data</td>
<td>AWS SCT supports conversion of spatial data from MS SQL Server to PostgreSQL.</td>
</tr>
<tr>
<td>MS SQL Server to MySQL: Spatial data</td>
<td>AWS SCT supports conversion of spatial data from MS SQL Server to MySQL.</td>
</tr>
<tr>
<td>MS SQL Server to PostgreSQL: @@ROWCOUNT</td>
<td>AWS SCT supports the Microsoft SQL Server @@ROWCOUNT function from MS SQL Server to PostgreSQL.</td>
</tr>
<tr>
<td>Oracle to MySQL: Formatting strings for conversion function TO_CHAR</td>
<td>The Oracle TO_CHAR function accepts a wide range of formatting characters. AWS SCT supports these formatting strings from Oracle to MySQL.</td>
</tr>
<tr>
<td>Oracle to MySQL: WITH clause support for INSERT and UPDATE statements</td>
<td>AWS SCT supports INSERT and UPDATE statements that use the WITH clause, from Oracle to MySQL.</td>
</tr>
<tr>
<td>Oracle to MySQL: Formatting strings for conversion function TO_NUMBER</td>
<td>The Oracle TO_NUMBER function accepts a wide range of formatting characters. AWS SCT supports these formatting strings from Oracle to MySQL.</td>
</tr>
<tr>
<td>MS SQL Server to MySQL: sp_sequence_get_range</td>
<td>The Microsoft SQL Server sp_sequence_get_range function returns a range of sequence values from a sequence object. AWS SCT supports this function from MS SQL Server to MySQL.</td>
</tr>
<tr>
<td>MS SQL Server to PostgreSQL: PARSE</td>
<td>AWS SCT supports the Microsoft SQL Server PARSE function, from MS SQL Server to PostgreSQL.</td>
</tr>
<tr>
<td>MS SQL Server - Saving SQL for source tree creation</td>
<td>You can save the SQL statements that AWS SCT generates to create the source tree.</td>
</tr>
</tbody>
</table>

### Issues Resolved

- SCT data extractors now emit COPY commands for Redshift.
- Redshift: count(distinct) window function need to re-write to Redshift DENSE_RANK.
- Conversion of queries using the Oracle FROM_TZ and EXTRACT functions are now correctly translated to Redshift-based equivalents, using extract and convert_timezone.
• Assessment Report inconsistently reporting number of code objects on the source DB.
• Re-write of Oracle INSTR function to Redshift built in function for more cases.
• Added PostgreSQL 9.x - PostgreSQL 10 schema compare support.
Release Notes for the AWS Schema Conversion Tool Build 617

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.617.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle to Amazon RDS for Oracle - Oracle server level objects</td>
<td>AWS SCT adds support for user profiles, user roles, event schedules, and more. For more information, see [Converting Oracle to Amazon RDS for Oracle](p. 40).</td>
</tr>
<tr>
<td>SQL Server to Amazon RDS for SQL Server - Assessment report includes additional metrics</td>
<td>The AWS SCT assessment report includes information on the Amazon RDS DB instance as well as SQL Server services in use by the source database. For more information, see [Converting SQL Server to Amazon RDS for SQL Server](p. 51).</td>
</tr>
<tr>
<td>Oracle to Amazon RDS for Oracle - Assessment report includes additional metrics</td>
<td>The AWS SCT assessment report includes information on the Amazon RDS DB instance as well as Oracle services in use by the source database. For more information, see [Converting Oracle to Amazon RDS for Oracle](p. 40).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL 10 - timestamp without time zone columns</td>
<td>AWS SCT supports timestamp without time zone columns.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL 10 - Row ID columns</td>
<td>You can convert ROWID pseudocolumns to data columns. For more information, see [Converting Oracle ROWID to PostgreSQL](p. 34).</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - Merge statement emulation</td>
<td>AWS SCT converts the MERGE statement when migrating to PostgreSQL. For more information, see [Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool](p. 43).</td>
</tr>
<tr>
<td>Oracle to MySQL - SELECT and WITH clause</td>
<td>AWS SCT converts a WITH clause when migrating to MySQL. For more information, see [Converting Oracle to Amazon RDS for MySQL or Amazon Aurora (MySQL)](p. 36).</td>
</tr>
<tr>
<td>SQL Server to MySQL - Extension pack</td>
<td>The AWS SCT extension pack emulates several functions including ISDATE, FORMAT, PATINDEX, and CONVERT.</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - user-defined table types</td>
<td>You can use user-defined table types to specify table structures. For more information, see [Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool](p. 43).</td>
</tr>
<tr>
<td>Oracle - Saving SQL for source tree creation</td>
<td>You can save the SQL statements that AWS SCT generates to create the source tree.</td>
</tr>
</tbody>
</table>

**Issues Resolved**

- Improvements to the assessment report for server-level objects.
- Fixed incorrect processing of objects ending with END block.
- Added assessment report generation date in the PDF file.
- Added logic to script generation for multiple files on Save as SQL option.
## Release Notes for the AWS Schema Conversion Tool Build 616

The following table shows the features and bug fixes for the AWS Schema Conversion Tool version 1.0.616.

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SQL Server to PostgreSQL - SUSER_SNAME support</strong></td>
<td>AWS SCT adds support for converting SUSER_SNAME. For more information, see Converting SQL Server to PostgreSQL (p. 48).</td>
</tr>
<tr>
<td><strong>SQL Server to PostgreSQL - Table-valued functions support</strong></td>
<td>AWS SCT adds support for converting table-valued functions. For more information, see Converting SQL Server to PostgreSQL (p. 48).</td>
</tr>
<tr>
<td><strong>Oracle to Amazon RDS for Oracle - Oracle jobs</strong></td>
<td>AWS SCT adds support for many types of Oracle jobs in Amazon RDS for Oracle.</td>
</tr>
<tr>
<td><strong>Oracle to Amazon RDS for Oracle - Oracle RAC</strong></td>
<td>Amazon RDS for Oracle doesn't support Oracle RAC. Consider using a Multi-AZ deployment on an Amazon RDS instance for high availability.</td>
</tr>
<tr>
<td><strong>Oracle to Amazon RDS for Oracle - Oracle Data Guard and Active Data Guard</strong></td>
<td>Amazon RDS for Oracle doesn't support Oracle Data Guard and Active Data Guard. Consider using a Multi-AZ deployment on an Amazon RDS instance for high availability.</td>
</tr>
<tr>
<td><strong>Oracle to Amazon RDS for Oracle - ongoing replication</strong></td>
<td>Amazon RDS for Oracle doesn't support ongoing replication. You can use AWS Database Migration Service if you need to have ongoing replication to a target on Amazon RDS.</td>
</tr>
<tr>
<td><strong>Oracle to Amazon RDS for Oracle - auditing</strong></td>
<td>Amazon RDS for Oracle doesn't support Oracle Unified Auditing. Amazon RDS for Oracle supports traditional auditing and fine-grained auditing (DBMS_FGA package).</td>
</tr>
<tr>
<td><strong>Oracle to Amazon RDS for Oracle - schedule objects</strong></td>
<td>AWS SCT supports converting Oracle DBMS_SCHEDULER objects when migrating to Amazon RDS for Oracle.</td>
</tr>
<tr>
<td><strong>SQL Server to MySQL - table-valued functions</strong></td>
<td>MySQL doesn't support multi-statement table-valued functions. AWS SCT simulates table-valued functions during a conversion by creating temporary tables. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (p. 43).</td>
</tr>
</tbody>
</table>
| **AWS SCT assessment report updates** | The AWS SCT assessment report updates include the following:  
  - Shows the size of databases on an Amazon RDS DB instance  
  - Shows the amount of storage used on an Amazon RDS DB instance  
  - Shows the number of databases on an Amazon RDS DB instance  
  - Shows if databases included in a migration are already being used for replication |
<p>| <strong>SQL Server to Amazon RDS for SQL Server - Service Broker and endpoints</strong> | Amazon RDS currently does not support Service Broker or additional T-SQL endpoints that use the CREATE ENDPOINT command. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (p. 43). |</p>
<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Db2 LUW to PostgreSQL 10 - partitioned tables</td>
<td>AWS SCT can convert Db2 LUW tables to partitioned tables in PostgreSQL 10.</td>
</tr>
</tbody>
</table>

Issues Resolved

- Db2 LUW to Amazon RDS for MySQL or Amazon Aurora (MySQL). Added ability to show target SQL for several transformed related objects.
- Db2 LUW to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL). Added ability to show target SQL for several transformed related objects.
- MySQL to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL). Added ability to show target SQL for several transformed related objects.
- OLAP. Added ability to show target SQL for several transformed related objects.
- AWS SCT issue with MySQL to MySQL conversion fixed.
Release Notes for the AWS Schema Conversion Tool Build 615

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

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Server to MySQL - Support for MERGE statement</td>
<td>MySQL does not support the MERGE statement, but AWS SCT can emulate the statement by using the INSERT ON DUPLICATE KEY clause and the UPDATE FROM and DELETE FROM statements. For more information, see Converting SQL Server to MySQL (p. 46).</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - Support for creating unique index names</td>
<td>AWS SCT gives you the option of generating unique index names if your index names are not unique. To do this, choose the option Generate unique index names in the project properties. For more information, see Converting SQL Server to PostgreSQL (p. 48).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL - Prevent overlapping database sequence values</td>
<td>For Oracle to PostgreSQL migration projects, choose the option Populate converted sequences with the last values generated on the source side in the Conversion settings tab of Project Settings. For more information, see Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL) (p. 33).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL - Support for PostgreSQL 10 partitioning</td>
<td>AWS SCT can emulate partitions and subpartitions when converting a schema from an Oracle database to a PostgreSQL database. For more information, see Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL) (p. 33).</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - Support for PostgreSQL 10 partitioning</td>
<td>AWS SCT can emulate partitions and subpartitions when converting a schema from an SQL Server database to a PostgreSQL database. For more information, see Converting SQL Server to PostgreSQL (p. 48).</td>
</tr>
<tr>
<td>SQL Server to MySQL - Support for GOTO statement</td>
<td>MySQL does not use a GOTO statement. When AWS SCT converts code that contains the GOTO statement, it converts the statement to use a BEGIN...END or LOOP...END LOOP statement.</td>
</tr>
<tr>
<td>SQL Server to PostgreSQL - Support for GOTO statement</td>
<td>PostgreSQL does not use a GOTO statement. When AWS SCT converts code that contains the GOTO statement, it converts the statement to use a BEGIN...END or LOOP...END LOOP statement.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL - Support for GOTO statement</td>
<td>PostgreSQL does not use a GOTO statement. When AWS SCT converts code that contains the GOTO statement, it converts the statement to use a BEGIN...END or LOOP...END LOOP statement.</td>
</tr>
<tr>
<td>DB2 LUW to PostgreSQL - Support for triggers from DB2 to PostgreSQL</td>
<td>AWS SCT can convert the various TRIGGER statements used with DB2 LUW. For more information, see Converting DB2 LUW to Amazon RDS for PostgreSQL or Amazon Aurora (PostgreSQL) (p. 60)</td>
</tr>
<tr>
<td>SQL Server to Amazon RDS for SQL Server - Support for database level triggers</td>
<td>AWS SCT can add database triggers to the object tree when Amazon RDS for SQL Server is the target.</td>
</tr>
<tr>
<td>SQL Server to Amazon RDS for SQL Server - Support for server-level triggers, linked</td>
<td>AWS SCT now supports server-level triggers, linked servers, and SQL Server Agents. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (p. 43).</td>
</tr>
<tr>
<td>New Feature or Enhancement</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>servers, and SQL Server Agents</td>
<td>AWS SCT can add directory objects to the object tree. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (p. 43).</td>
</tr>
<tr>
<td>Oracle to Amazon RDS for Oracle - Support for directory objects, tablespaces, and user roles and privileges</td>
<td>AWS SCT now supports IBM Db2 LUW version 10.1.</td>
</tr>
<tr>
<td>Added support of isolated government regions</td>
<td>AWS SCT now supports isolated government regions.</td>
</tr>
</tbody>
</table>

Issues Resolved

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

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

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle to PostgreSQL: Assessment Report for SQL*Plus Conversion</td>
<td>AWS SCT can convert SQL<em>Plus files into PSQL. The assessment report shows how AWS SCT converted the SQL</em>Plus files into PSQL. For more information, see [Using Oracle as a Source for AWS Schema Conversion Tool](p. 30).</td>
</tr>
<tr>
<td>Aurora MySQL Compatible 5.7 version support</td>
<td>Added support for converting Aurora MySQL 5.7 schemas.</td>
</tr>
<tr>
<td>Db2 LUW 9.1</td>
<td>Added support for Db2 LUW version 9.1.</td>
</tr>
<tr>
<td>OLTP Data Migration. Data compression.</td>
<td>Data compression during migration is now optional when you set up a task using a Replication agent.</td>
</tr>
<tr>
<td>Oracle to Oracle RDS: DB Links support</td>
<td>Oracle to Oracle RDS migrations now support DB Links. For more information, see [Using Oracle as a Source for AWS Schema Conversion Tool](p. 30).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL: SELECT INTO BULK COLLECT (VARRAY) conversion</td>
<td>SQL statements using BULK COLLECT (VARRAY) can now be converted when migrating between Oracle and PostgreSQL.</td>
</tr>
<tr>
<td>Oracle comments conversion</td>
<td>Added support for converting Oracle comments into the format used by the target database engine. For more information, see [Using Oracle as a Source for AWS Schema Conversion Tool](p. 30).</td>
</tr>
<tr>
<td>Emulation of Oracle system objects</td>
<td>Added support for converting Oracle system objects into PostgreSQL. For more information, see [Using Oracle as a Source for AWS Schema Conversion Tool](p. 30).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL: ROWNUM converted to LIMIT</td>
<td>Added support for converting ROWNUM.</td>
</tr>
<tr>
<td>Microsoft SQL Server to Microsoft SQL Server RDS: BULK INSERT and OPENROWSET()</td>
<td>Added support converting BULK INSERT and OPENROWSET()</td>
</tr>
<tr>
<td>Microsoft SQL Server to Microsoft SQL Server RDS: Links inside storage objects</td>
<td>AWS SCT now supports links inside stored objects during a migration to Amazon RDS. For more information, see [Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool](p. 43).</td>
</tr>
<tr>
<td>Microsoft SQL Server to PostgreSQL: PATINDEX</td>
<td>AWS SCT now supports converting PATINDEX during a migration to PostgreSQL. For more information, see [Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool](p. 43).</td>
</tr>
<tr>
<td>Microsoft SQL Server to PostgreSQL: System objects access</td>
<td>SQL Server system object are now converted to objects in PostgreSQL. For more information, see [Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool](p. 43).</td>
</tr>
</tbody>
</table>
### New Feature or Enhancement

| Microsoft SQL Server to PostgreSQL: Inline function creation | Added support for inline functions. For more information, see Using Microsoft SQL Server as a Source for AWS Schema Conversion Tool (p. 43). |

### Issues Resolved

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

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

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 LUW support</td>
<td>Added support for migrating a DB2 database as a source. For more information, see [Converting Database Schemas Using the AWS Schema Conversion Tool](p. 92).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion. Added SQL*Plus files conversion</td>
<td>You can now convert SQL*Plus files.</td>
</tr>
<tr>
<td>Snowball Tab added for OLTP Data Migration</td>
<td>A new Snowball tab was added for OLTP Data Migration that shows you the current status of a Snowball device for particular project. For more information, see [Using Data Extraction Agents](p. 135).</td>
</tr>
<tr>
<td>Oracle To PostgreSQL Conversion. Convert Oracle spatial code to PostGIS open</td>
<td>Oracle To PostgreSQL Conversion. Oracle spatial code converts to PostGIS.</td>
</tr>
<tr>
<td>Schema Compare Added Oracle 10 Support</td>
<td>You can now run Schema Compare with Oracle 10. For more information, see [Comparing Database Schemas](p. 104).</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion. Implicit typecasting of Oracle db handled in PostgreSQL</td>
<td>While converting from Oracle to PostgreSQL, SCT add data type casting.</td>
</tr>
<tr>
<td>SQL Server To PostgreSQL Conversion</td>
<td>Added CITEXT type support. Now you can choose it with data type mapping.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion</td>
<td>Additional improvements of Dynamic SQL Conversion for EXECUTE IMMEDIATE and DBMS_SQL and Cursors.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion</td>
<td>Added support of SELECT INTO BULK COLLECT Conversion for Oracle to PostgreSQL.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion</td>
<td>Now it is possible to convert an Associative array from Oracle to PostgreSQL.</td>
</tr>
<tr>
<td>Type Mapping - Custom type mapping improvements</td>
<td>Added ability to select source data types based on length and precision.</td>
</tr>
<tr>
<td>AWS Profile Settings</td>
<td>Added ability to select default profile in AWS Profile Settings.</td>
</tr>
<tr>
<td>Greenplum to Redshift Conversion</td>
<td>BuiltIn SQL Functions to Redshift Scalar SQL UDF conversion added.</td>
</tr>
</tbody>
</table>
### Issues Resolved

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

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

<table>
<thead>
<tr>
<th>New Feature or Enhancement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom Data Type Mapping</td>
<td>You can now set up mapping rules to change the data type for storage objects. You define what data type should be changed in Mapping Rules tab.</td>
</tr>
<tr>
<td>What's New option added</td>
<td>A new &quot;What's New&quot; option under the Help menu shows you all major features that were added for this release.</td>
</tr>
<tr>
<td>Schema Compare Added Oracle 10 Support</td>
<td>You can now run Schema Compare with Oracle 10.</td>
</tr>
<tr>
<td>Oracle to PostgreSQL Conversion.</td>
<td>The migration from Oracle to PostgreSQL now supports</td>
</tr>
<tr>
<td></td>
<td>• global variables</td>
</tr>
<tr>
<td></td>
<td>• associative arrays</td>
</tr>
<tr>
<td></td>
<td>• formatting strings for TO_NUMBER function</td>
</tr>
<tr>
<td></td>
<td>• converting Dynamic SQL with DBMS_SQL package</td>
</tr>
<tr>
<td></td>
<td>• converting multiple nested subprograms with global variables</td>
</tr>
<tr>
<td>Added support for loading some PostgreSQL</td>
<td>Added support for loading PostgreSQL IMMUTABLE, STABLE, and VOLATILE function attributes.</td>
</tr>
<tr>
<td>function attributes and domain constraints</td>
<td>Added support for loading PostgreSQL domain constraints.</td>
</tr>
</tbody>
</table>

**Issues Resolved**

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

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

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

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>more information, see <a href="#">Using Virtual Partitioning with AWS Schema Conversion Tool (p. 154)</a>.</td>
</tr>
</tbody>
</table>

## Issues Resolved

<table>
<thead>
<tr>
<th>Date reported</th>
<th>Description</th>
</tr>
</thead>
</table>
| post version 608 | • OLTP Data Migration. Added the ability to refresh Agent/Task Logs.  
• Oracle to Oracle schema compare. Fixed the comparison of public synonyms.  
• Fixed the ability to hide system schema from Tree View.  
• Added a Tooltip for the tree filter. Allows user to upload .csv files with objects that needs to be filtered.  
• Fixed conversion of Oracle NUMTOINTERVAL to Redshift interval literal where applicable.  
• Oracle to Redshift Conversion. Fixed the migration of a Redshift SUBSTR with a second parameter constant (start of substr) to be a simple expression rather than a CASE statement.  
• OLAP Data Migration. Added description for virtual partitioning. |
The following table describes the important changes to the AWS Schema Conversion Tool 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 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 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>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.</td>
<td>April 26, 2019</td>
</tr>
<tr>
<td>Build</td>
<td>Updates</td>
<td>Date</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>#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>#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 Oracle extension during Oracle conversion to PostgreSQL.</td>
<td>February 22, 2019</td>
</tr>
<tr>
<td>#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>#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>#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>#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>#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>-----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------</td>
<td>---------------</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>
<tr>
<td>AWS SCT build #1.0.611</td>
<td>Updates include support for Oracle to PostgreSQL dynamic statements, opening the log file by selecting an error message, and the ability to hide schemas in tree view.</td>
<td>January 23, 2018</td>
</tr>
</tbody>
</table>

**Earlier Updates**

The following table describes the important changes to the AWS Schema Conversion Tool 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. 14).</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. 14).</td>
<td>October 30, 2017</td>
</tr>
<tr>
<td>1.0.607</td>
<td>Data extraction tasks can ignore LOBs</td>
<td>When you create data extraction tasks, you can now choose to ignore large objects (LOBs) to reduce the amount of data that you extract. For more information, see Creating, Running, and Monitoring an AWS SCT Data Extraction Task (p. 145).</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. 145).</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 Extraction Filters in the AWS Schema Conversion Tool (p. 143).</td>
<td>June 16, 2017</td>
</tr>
<tr>
<td>1.0.603</td>
<td>AWS SCT supports additional data warehouse versions</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 information, see Converting Data Warehouse Schemas to Amazon Redshift Using the AWS Schema Conversion Tool (p. 106).</td>
<td>June 16, 2017</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. 135).</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</td>
<td>May 11, 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>1.0.601</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 the AWS Schema Conversion Tool](p. 106).</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. 135).</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. 138).</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. 153).</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. 135).</td>
<td>February 16, 2017</td>
</tr>
<tr>
<td>1.0.600</td>
<td>Customer Feedback</td>
<td>You can now provide feedback about the AWS Schema Conversion Tool. You can file a bug report, you can submit a feature request, or you can provide general information. For more information, see [Providing Customer Feedback](p. 3).</td>
<td>February 16, 2017</td>
</tr>
<tr>
<td>1.0.502</td>
<td>Integration with AWS DMS</td>
<td>You can now use the AWS Schema Conversion Tool to create AWS DMS endpoints and tasks. You can run and monitor the tasks from AWS SCT. For more information, see [Using the AWS Schema Conversion Tool with the AWS Database Migration Service](p. 134).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>1.0.502</td>
<td>Amazon Aurora with PostgreSQL compatibility as a target database</td>
<td>The AWS Schema Conversion Tool now supports Amazon Aurora with PostgreSQL compatibility as a target database. For more information, see [Converting Database Schemas Using the AWS Schema Conversion Tool](p. 92).</td>
<td>December 20, 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>1.0.502</td>
<td>Support for profiles</td>
<td>You can now store different profiles in the AWS Schema Conversion Tool and easily switch between them. For more information, see Using AWS Service Profiles in the AWS Schema Conversion Tool (p. 13).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>1.0.501</td>
<td>Support for Greenplum Database and Netezza</td>
<td>You can now use the AWS Schema Conversion Tool to convert your data warehouse schemas from Greenplum Database and Netezza to Amazon Redshift. For more information, see Converting Data Warehouse Schemas to Amazon Redshift Using the AWS Schema Conversion Tool (p. 106).</td>
<td>November 17, 2016</td>
</tr>
<tr>
<td>1.0.501</td>
<td>Redshift optimization</td>
<td>You can now use the AWS Schema Conversion Tool to optimize your Amazon Redshift databases. For more information, see Optimizing Amazon Redshift by Using the AWS Schema Conversion Tool (p. 121).</td>
<td>November 17, 2016</td>
</tr>
<tr>
<td>1.0.500</td>
<td>Mapping rules</td>
<td>Before you convert your schema with the AWS Schema Conversion Tool, you can now set up rules that change the data type of columns, move objects from one schema to another, and change the names of objects. For more information, see Creating Mapping Rules in the AWS Schema Conversion Tool (p. 109).</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 Assessment Reports with AWS Schema Conversion Tool (p. 77).</td>
<td>October 4, 2016</td>
</tr>
<tr>
<td>1.0.400</td>
<td>Data warehouse schema conversions</td>
<td>You can now use the AWS Schema Conversion Tool to convert your data warehouse schemas from Oracle and Teradata to Amazon Redshift. For more information, see Converting Data Warehouse Schemas to Amazon Redshift Using the AWS Schema Conversion Tool (p. 106).</td>
<td>July 13, 2016</td>
</tr>
<tr>
<td>1.0.400</td>
<td>Application SQL conversions</td>
<td>You can now use the AWS Schema Conversion Tool to convert SQL in your C++, C#, Java, or other application code. For more information, see Converting Application SQL Using the AWS Schema Conversion Tool (p. 180).</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>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. 188) and Using the Custom Python Library for the AWS SCT Extension Pack (p. 187).</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>