aws

用户指南

AWS Schema Conversion Tool

版本 1.0.672

AWS Schema Conversion Tool: 用户指南

Table of Contents

AWS Schema Conversion Tool 是什么？

您可以使用 AWS Schema Conversion Tool（AWS SCT）将现有的数据库架构从一个数据库引擎转换为另一个数据库引擎。您可以转换关系 OLTP 架构或数据仓库架构。转换后的 Schema 适用于 Amazon Relational Database Service（Amazon RDS）MySQL、MariaDB、Oracle、SQL Server、PostgreSQL 数据库、Amazon Aurora 数据库集群或Amazon Redshift 集群。转换后的架构也可用于 Amazon EC2 实例上的数据库或作为数据存储在 Amazon S3 存储桶中。

AWS SCT 支持多种行业标准，包括美国联邦信息处理标准（FIPS），用于连接到 Amazon S3 存储桶或其他 AWS 资源。AWS SCT 也符合美国联邦风险与授权管理计划（FedRAMP）的要求。有关 AWS 和合规性工作的详细信息，请参阅 AWS 按合规性计划提供的范围内服务。

AWS SCT 支持以下 OLTP 转换。

源数据库	目标数据库
IBM Db2 for z/OS（版本 12）	Amazon Aurora MySQL 兼容版（Aurora MySQL）、Amazon Aurora PostgreSQL 兼容版（Aurora PostgreSQL）、MySQL、PostgreSQL 有关更多信息，请参阅将 IBM Db2 for z/OS 用作源。
IBM Db2 LUW（版本 9.1、9.5、9.7、10.5、11.1 和 11.5）	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL 有关更多信息，请参阅将 IBM Db2 LUW 作为源。
Microsoft Azure SQL 数据库	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL 有关更多信息，请参阅将 Azure SQL 数据库作为源。
Microsoft SQL Server（版本 2008 R2、2012、2014、2016、2017、2019 和 2022）。	Aurora MySQL, Aurora PostgreSQL、适用于 Aurora PostgreSQL 的 Babelfish（仅用于评估

源数据库	目标数据库
	报告）、MariaDB、Microsoft SQL Server、MySQL, PostgreSQL 有关更多信息，请参阅将 SQL Server 作为源。
MySQL（版本 5.5 及更高版本）	Aurora PostgreSQL、MySQL、PostgreSQL 有关更多信息，请参阅将 MySQL 作为源。 您可以将架构和数据从 MySQL 迁移到 Aurora MySQL 数据库集群，而无需使用 AWS SCT。有关更多信息，请参阅将数据迁移到 Amazon Aurora 数据库集群。
Oracle（版本 10.1 及更高版本）	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、Oracle、PostgreSQL 有关更多信息，请参阅将 Oracle 数据库作为源。
PostgreSQL（版本 9.1 及更高版本）	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL 有关更多信息，请参阅将 PostgreSQL 作为源。
SAP ASE（版本 12.5.4、15.0.2、15.5、15.7 和 16.0）	Aurora MySQL、Aurora PostgreSQL、MariaDB, MySQL、PostgreSQL 有关更多信息，请参阅使用 SAP ASE (Sybase ASE) 作为源。

AWS SCT 支持以下数据仓库转换。

源数据仓库	目标数据仓库
Amazon Redshift	Amazon Redshift

源数据仓库	目标数据仓库
	有关更多信息，请参阅将 Amazon Redshift 用作源。
Azure Synapse Analytics	Amazon Redshift 有关更多信息，请参阅使用 Azure Synapse Analytics 作为源。
BigQuery	Amazon Redshift 有关更多信息，请参阅使用 BigQuery 作源。
Greenplum 数据库（版本 4.3 和 6.21）	Amazon Redshift 有关更多信息，请参阅将 Greenplum 数据库用作源。
Microsoft SQL Server（版本 2008 及更高版本）	Amazon Redshift 有关更多信息，请参阅将 SQL Server 数据仓库用作源。
Netezza（版本 7.0.3 及更高版本）	Amazon Redshift 有关更多信息，请参阅将 Netezza 用作源。
Oracle（版本 10.1 及更高版本）	Amazon Redshift 有关更多信息，请参阅将 Oracle 数据仓库用作源。
Snowflake（版本 3）	Amazon Redshift 有关更多信息，请参阅使用 Snowflake 作为源。
Teradata（版本 13 及更高版本）	Amazon Redshift 有关更多信息，请参阅将 Teradata 用作源。

源数据仓库	目标数据仓库
Vertica（版本 7.2.2 及更高版本）	Amazon Redshift 有关更多信息，请参阅将 Vertica 用作源。

AWS SCT 支持以下数据 NoSQL 数据库转换。

源数据库	目标数据库
Apache Cassandra（版本 2.1.x、2.2.16 和 3.11.x）	Amazon DynamoDB 有关更多信息，请参阅使用 Apache Cassandra 作为源。

AWS SCT 支持以下提取、转换和加载（ETL）流程的转换。有关更多信息，请参阅转换 ETL 过程。

源	目标
Informatica ETL 脚本	Informatica
Microsoft SQL Server Integration Services (SSIS) ETL 包	AWS Glue 或 AWS Glue Studio
带有来自 Teradata Basic Teradata Query (BTEQ) 的嵌入式命令的 Shell 脚本	Amazon Redshift SQL
Teradata BTEQ ETL 脚本	AWS Glue 或 Amazon Redshift RSQL
Teradata FastExport 作业脚本	Amazon Redshift SQL
Teradata FastLoad 作业脚本	Amazon Redshift SQL
Teradata MultiLoad 作业脚本	Amazon Redshift SQL

AWS SCT 支持以下大数据框架迁移。有关更多信息，请参阅迁移大数据框架。

源	目标
Apache Hive（版本 0.13.0 及更高版本）	Amazon EMR 的 Hive
Apache HDFS	Amazon S3 或 Amazon EMR 的 HDFS
Apache Oozie	AWS Step Functions

架构转换概述

AWS SCT 提供基于项目的用户界面，以便将源数据库的数据库架构自动转换为与目标 Amazon RDS 实例兼容的格式。如果无法自动转换源数据库中的架构，AWS SCT 将提供有关如何在目标 Amazon RDS 数据库中创建等效架构的指导。

有关如何安装 AWS SCT 的信息，请参阅安装、验证和更新 AWS SCT。

有关 AWS SCT 用户界面的简介，请参阅 AWS SCT 用户界面的使用。

有关转换过程的信息，请参阅使用 AWS SCT 转换数据库架构。

除了将一个数据库引擎中的现有数据库架构转换到另一个引擎，AWS SCT 还提供了一些其他功能，可帮助您将数据和应用程序迁移到 AWS 云上：

- 您可以使用数据提取代理从数据仓库中提取数据，以准备将其迁移到 Amazon Redshift。要管理数据提取代理，您可以使用 AWS SCT。有关更多信息，请参阅将本地数据仓库中的数据迁移到 Amazon Redshift。
- 您可以使用 AWS SCT 创建 AWS DMS 终端节点和任务。您可以从 AWS SCT 中运行和监控这些任务。有关更多信息，请参阅配合使用 AWS SCT 和 AWS DMS。
- 在某些情况下，数据库功能无法转换为等效的 Amazon RDS 或 Amazon Redshift 功能。AWS SCT 扩展包向导可以帮助您安装 AWS Lambda 函数和 Python 库来模拟无法转换的功能。有关更多信息，请参阅使用 AWS SCT 扩展包。
- 您可以使用 AWS SCT 优化现有 Amazon Redshift 数据库。AWS SCT 建议使用排序键和分配键优化您的数据库。有关更多信息，请参阅使用 AWS SCT 优化 Amazon Redshift。
- 您可以使用 AWS SCT 将现有本地数据库架构复制到运行相同引擎的 Amazon RDS 数据库实例。您可以使用此功能来分析迁移到云和更改许可证类型的潜在成本节省。
- 您可以使用 AWS SCT 通过 C++、C#、Java 或其他应用程序代码转换 SQL。您可以查看、分析、编辑和保存转换后的 SQL 代码。有关更多信息，请参阅使用 AWS SCT 转换应用程序 SQL。

- 您可以使用 AWS SCT 迁移提取、转换和加载（ETL）流程。有关更多信息，请参阅使用 AWS Schema Conversion Tool 转换提取、转换和加载（ETL）过程。

提供反馈

您可以提供关于 AWS SCT 的反馈。您可以提交错误报告、提交功能请求，也可以提供常规信息。

提供关于 AWS SCT 的反馈。

1. 启动 AWS Schema Conversion Tool。
2. 打开 Help 菜单，然后选择 Leave Feedback。随即出现 Leave Feedback 对话框。
3. 对于 Area，选择 Information、Bug report 或 Feature request。
4. 对于 Source database，选择您的源数据库。如果您的反馈并不是针对特定数据库的，请选择 Any。
5. 对于 Target database，选择您的目标数据库。如果您的反馈并不是针对特定数据库的，请选择 Any。
6. 对于 Title，请输入反馈的标题。
7. 对于 Message ，请输入您的反馈。
8. 选择 Send 以提交您的反馈。

安装、验证和更新 AWS SCT

AWS Schema Conversion Tool (AWS SCT) 是一个独立的应用程序，提供基于项目的用户界面。AWS SCT 适用于微软 Windows、Fedora Linux 和 Ubuntu Linux。 AWS SCT 仅在 64 位操作系统上支持。

为确保您获得正确版本的 AWS SCT 分发文件，我们会在您下载压缩文件后提供验证步骤。您可以使用提供的步骤验证文件。

AWS SCT 既可以作为独立应用程序使用，也可以作为命令行工具使用。有关命令行工具的信息，请参见AWS SCT CLI。

主题

正在安装 AWS SCT

您可以安装 AWS SCT 在以下操作系统上：

- Microsoft Windows 10
- Fedora Linux 36 及更高版本
- Ubuntu Linux 18 及更高版本

要安装 AWS SCT

1. 使用您的操作系统的链接，下载包含 AWS SCT 安装程序的压缩文件。所有压缩文件的扩展名均为 .zip。解压缩 AWS SCT 安装程序文件时，其格式将与您的操作系统相匹配。

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

2. 解压缩操作系统的 AWS SCT 安装程序文件，如下所示。

操作系统	文件名
Fedora Linux	aws-schema-conversion-tool-1.0. *build-number* .x86_64.rpm
Microsoft Windows	AWS Schema Conversion Tool-1.0. *build-number* .msi
Ubuntu Linux	aws-schema-conversion-tool-1.0. *build-number* .deb

3. 运行上一步中提取的 AWS SCT 安装程序文件。使用适合您的操作系统的说明，如下所示。

操作系统	安装说明
Fedora Linux	在存储该下载文件的文件夹中运行以下命令： sudo yum install aws-schema-conversion-tool-1.0. *build-number* .x86_64.rpm
Microsoft Windows	双击该文件以运行安装程序。
Ubuntu Linux	在存储该下载文件的文件夹中运行以下命令： sudo dpkg -i aws-schema-conversion-tool-1.0. *build-number* .deb

4. 下载适用于您的源数据库引擎和目标数据库引擎的 Java 数据库连接（JDBC）驱动程序。有关说明和下载链接，请参阅下载所需的数据库驱动程序。

现在，您已经完成了 AWS SCT 应用程序的设置。双击该应用程序的图标运行 AWS SCT。

验证 AWS SCT 文件下载

有几种方法可以验证的分发文件 AWS SCT。最简单的方法是将该文件的校验和与 AWS已发布的校验和进行比较。为了提高安全性，您可以根据在其中安装文件的操作系统，使用以下过程验证分配文件。

本节包括以下主题。

主题

验证文件的校验和 AWS SCT

为了检测下载或存储 AWS SCT 压缩文件时可能引入的任何错误，您可以将文件校验和与提供的 AWS 值进行比较。 AWS 使用 SHA256 算法计算校验和。

使用校验和验证 AWS SCT 分发文件

1. 使用 “安装” 部分中的链接下载 AWS SCT 分发文件。有关更多信息，请参阅 正在安装 AWS SCT。
2. 下载名为 sha256Check.txt 的最新校验和文件。此文件包含最新 AWS SCT 版本的校验和。例如，此时显示文件，如下所示：

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

3. 在包含分配文件的目录中对您的操作系统运行 SHA256 验证命令。例如，在 Linux 中运行以下命令。

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

4. 将该命令的结果与 sha256Check.txt 文件中显示的值进行比较。如果校验和匹配，则可以放心运行分配文件。如果校验和不匹配，则不要运行分配文件，并联系 AWS Support。

验证 Fedora 上的 AWS SCT RPM 文件

AWS 除了分发文件校验和之外，还提供了另一个级别的验证。分发文件中的所有 RPM 文件均由 AWS 私钥签名。公有 GPG 密钥可在 amazon.com.public.gpg-key 上查看。

验证 Fedora 上的 AWS SCT RPM 文件

1. 使用 “安装” 部分中的链接下载 AWS SCT 分发文件。
2. 验证 AWS SCT 分发文件的校验和。
3. 提取分配文件的内容。找到要验证的 RPM 文件。
4. 从 amazon.com.public.gpg-key 下载 GPG 公有密钥。
5. 使用以下命令将该公有密钥导入到您的 RPM 数据库 (确保您有相应的权限)：

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

6. 运行以下命令检查导入是否成功：

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

7. 运行以下命令检查 RPM 签名：

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

验证 Ubuntu 上的 AWS SCT DEB 文件

AWS 除了分发文件校验和之外，还提供了另一个级别的验证。分配文件中的所有 DEB 文件均由 GPG 分离签名进行签名。

验证 Ubuntu 上的 AWS SCT DEB 文件

1. 使用 “安装” 部分中的链接下载 AWS SCT 分发文件。
2. 验证 AWS SCT 分发文件的校验和。
3. 提取分配文件的内容。找到要验证的 DEB 文件。
4. 从 aws-schema-conversion-tool-1.0. latest.deb.asc 下载分离的签名。
5. 从 amazon.com.public.gpg-key 下载 GPG 公有密钥。
6. 运行以下命令导入 GPG 公有密钥：

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

7. 运行以下命令验证签名：

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

在 AWS SCT 微软 Windows 上验证 MSI 文件

AWS 除了分发文件校验和之外，还提供了另一个级别的验证。MSI 文件有数字签名，你可以检查它是否由 AWS其签名。

在 Windows 上验证 AWS SCT MSI 文件

1. 使用 “安装” 部分中的链接下载 AWS SCT 分发文件。
2. 验证 AWS SCT 分发文件的校验和。
3. 提取分配文件的内容。找到要验证的 MSI 文件。
4. 在 Windows 资源管理器中，右键单击 MSI 文件并选择 Properties。
5. 选择数字签名选项卡。
6. 验证数字签名是否来自 Amazon Services LLC。

下载所需的数据库驱动程序

AWS SCT 要正常工作，请下载源数据库引擎和目标数据库引擎的 JDBC 驱动程序。如果您使用虚拟目标数据库平台，则无需为目标数据库引擎下载 JDBC 驱动程序。有关更多信息，请参阅 使用虚拟目标。

下载驱动程序后，指定驱动程序文件的位置。有关更多信息，请参阅 在全局设置中存储驱动程序路径。

您可以从以下位置下载数据库驱动程序。

> **Important**
>
> 下载最新版本的可用驱动程序。下表包括支持的最低版本的数据库驱动程序 AWS SCT。

数据库引擎	驱动程序	下载位置
Amazon Aurora MySQL 兼容版	mysql-connector-java-5.1.6.jar	https://www.mysql.com/products/connector/
Amazon Aurora PostgreSQL 兼容版	postgresql-42.2.19.jar	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar
Amazon EMR	HiveJDBC42.jar	http://awssupportdatasvcs.com/bootstrap-actions/Simba/latest/
Amazon Redshift	redshift-jdbc42-2.1.0.9.jar	https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.9/redshift-jdbc42-2.1.0.9.zip
Amazon Redshift Serverless	redshift-jdbc42-2.1.0.9.jar	https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.9/redshift-jdbc42-2.1.0.9.zip
Apache Hive	hive-jdbc-2.3.4-standalone.jar	https://repo1.maven.org/maven2/org/apache/hive/hive-jdbc/2.3.4/hive-jdbc-2.3.4-standalone.jar
Azure SQL 数据库	mssql-jdbc-7.2.2.jre11.jar	https://docs.microsoft.com/en-us/sql/connect/jdbc/ release-notes-for-the-jdbc-driver ? view= sql-server-ver 15 #72
Azure Synapse Analytics	mssql-jdbc-7.2.2.jre11.jar	https://docs.microsoft.com/en-us/sql/connect/jdbc/ release-notes-for-the-jdbc-driver ? view= sql-server-ver 15 #72
Greenplum 数据库	postgresql-42.2.19.jar	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar
IBM Db2 for z/OS	db2jcc-db2jcc4.jar	https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads-db2-zos

数据库引擎	驱动程序	下载位置
IBM Db2 LUW	db2jcc-db2jcc4.jar	https://www.ibm.com/support/pages/node/382667
MariaDB	mariadb-java-client-2.4.1.jar	https://downloads.mariadb.com/Connectors/java/connector-java-2.4.1/ mariadb-java-client-2.4.1.jar
Microsoft SQL Server	mssql-jdbc-10.2.jar	https://docs.microsoft.com/en-us/sql/connect/jdbc/ download-microsoft-jdbc-driver-for-sql-server ? view= 15 sql-server-ver
MySQL	mysql-connector-java-8.0.15.jar	https://dev.mysql.com/downloads/connector/j/
Netezza	nzjdbc.jar 使用客户端工具软件。下载驱动程序版本 7.2.1，该版本向后兼容数据仓库版本 7.2.0。	http://www.ibm.com/support/knowledgecenter/SSULQD_7.2.1/com.ibm.nz.datacon.doc/c_datacon_plg_overview.html
Oracle	ojdbc8.jar 支持版本 8 和更高版本的驱动程序。	https://www.oracle.com/database/technologies/jdbc-ucp-122-downloads.html
PostgreSQL	postgresql-42.2.19.jar	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar
SAP ASE (Sybase ASE)	jconn4.jar	jConnect JDBC 驱动程序
Snowflake	snowflake-jdbc-3.9.2.jar 有关更多信息，请参阅下载/集成 JDBC 驱动程序。	https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/3.9.2/snowflake-jdbc-3.9.2.jar

数据库引擎	驱动程序	下载位置

Teradata	`terajdbc4.jar`
	
	`tdgssconfig.jar`
	
	Teradata JDBC 驱动程序版本 16.20.00.11 及更高版不需要 `tdgssconfig.jar` 文件。
		https://downloads.teradata.com/download/connectivity/jdbc-driver

Vertica	`vertica-jdbc-9.1.1-0.jar` 支持版本 7.2.0 和更高版本的驱动程序。	https://www.vertica.com/client_drivers/9.1.x/9.1.1-0/vertica-jdbc-9.1.1-0.jar

在 Linux 上安装 JDBC 驱动程序

您可以使用以下步骤在 Linux 系统上安装 JDBC 驱动程序以供使用。 AWS SCT

在 Linux 系统上安装 JDBC 驱动程序

1. 创建一个目录来存储 JDBC 驱动程序。

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

2. 使用如下所示的命令安装适用于您的数据库引擎的 JDBC 驱动程序。

数据库引擎	安装命令
Amazon Aurora (兼容 MySQL)	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz`
Amazon Aurora (兼容 PostgreSQL)	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo cp -a /tmp/postgresql-X.X.X.jre7.tar .`

数据库引擎	安装命令
Microsoft SQL Server	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/sqljdbc_X.X.X_enu.tar.gz`

MySQL	`PROMPT> cd /usr/local/jdbc-drivers`
	`PROMPT> sudo tar xzvf /tmp/mysql-connector-java-X`
	`.X.X.tar.gz`

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

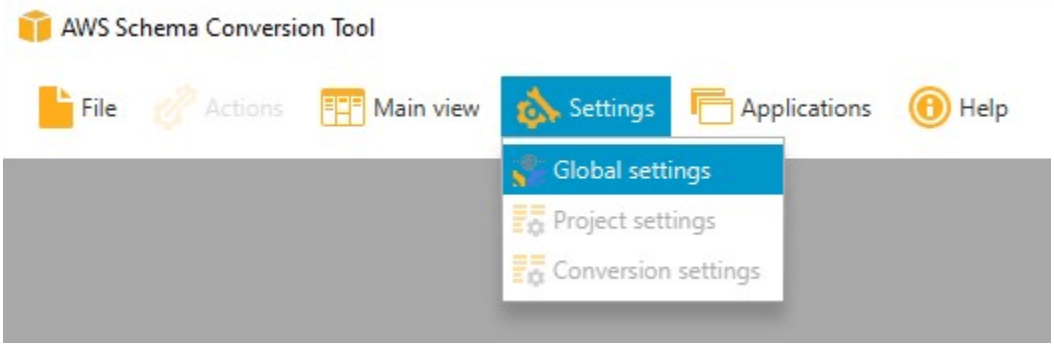
PostgreSQL	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo cp -a /tmp/postgresql-X.X.X.jre7.tar .`

在全局设置中存储驱动程序路径

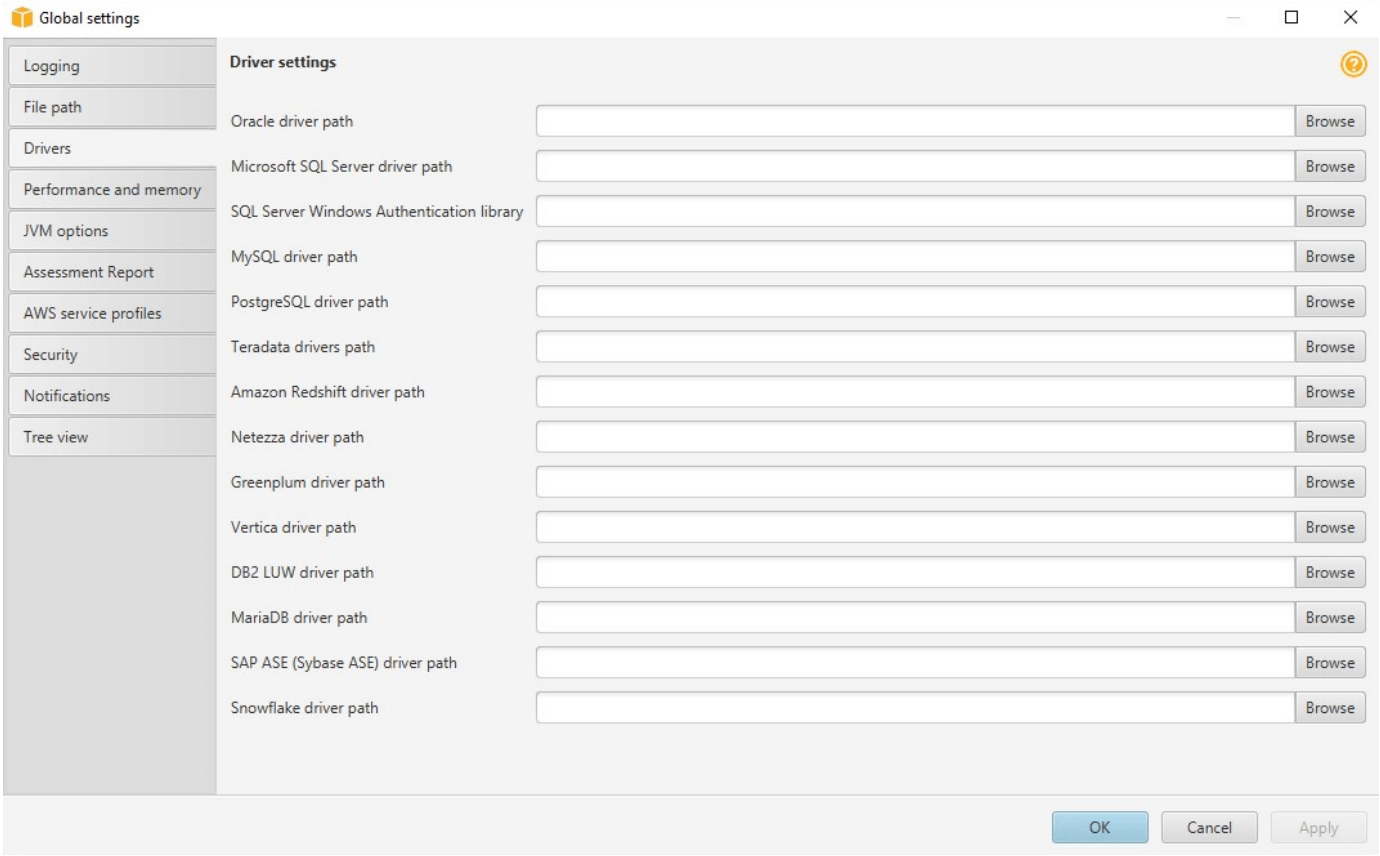
下载并安装所需的 JDBC 驱动程序后，可以在设置中全局设置驱动程序的位置。 AWS SCT 如果您未全局设置驱动程序的位置，当您连接到数据库时应用程序会要求您提供驱动程序的位置。

更新驱动程序文件位置

1. 在中 AWS SCT，选择 “设置”，然后选择 “全局设置”。



2. 对于全局设置，请选择 Drivers。添加适用于您的源数据库引擎和目标 Amazon RDS 数据库实例数据库引擎的 JDBC 驱动程序文件路径。



3. 添加完驱动程序路径之后，选择 OK。

正在更新 AWS SCT

AWS 定期更新 AWS SCT 新特性和功能。如果您要从以前的版本进行更新，请创建一个新 AWS SCT 项目并重新转换您正在使用的任何数据库对象。

您可以查看是否有更新 AWS SCT。

要查看更新 AWS SCT

1. 进入后 AWS SCT，选择 “帮助”，然后选择 “检查更新”。
2. 在 Check for Updates (检查更新) 对话框中，选择 What's New (新增功能)。如果此链接未出现，则表明您拥有最新版本。

AWS SCT CLI

您可以下载 AWS SCT CLI 以供命令行使用。要下载 JAR，请使用以下链接：

AWSSchemaConversionToolBatch.jar

AWS SCT 用户界面的使用

以下主题可帮助您使用 AWS SCT 用户界面。有关安装 AWS SCT 的信息，请参阅 安装、验证和更新 AWS SCT。

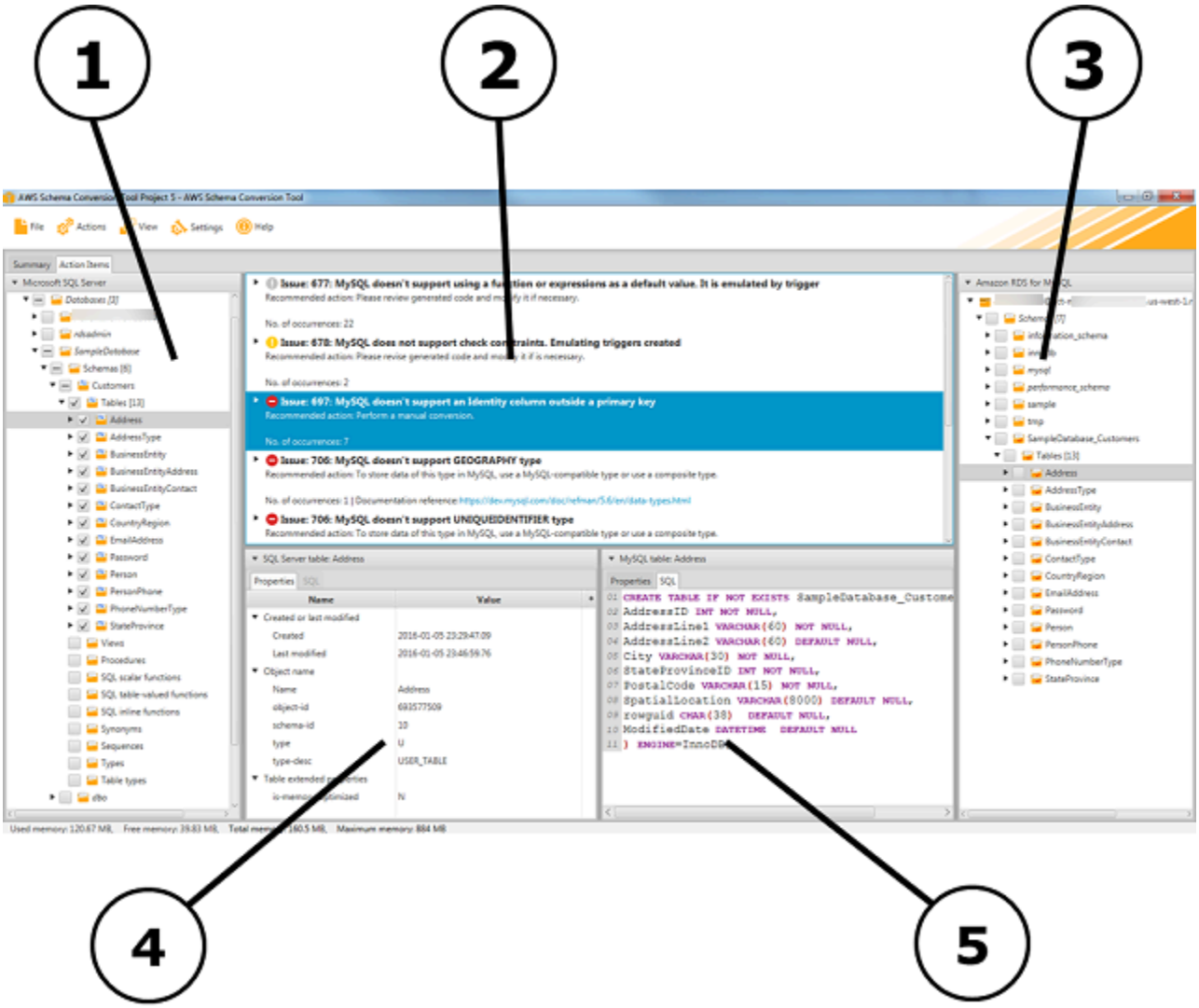
主题

AWS SCT 项目窗口

下图显示了当创建架构迁移项目，然后转换架构时在 AWS SCT 中所看到的内容。

1. 在左侧面板中，源数据库中的架构会显示在树状图中。您的数据库架构会“延迟加载”。换而言之，当您从树状图中选择某个项时，AWS SCT 会从您的源数据库中获取并显示当前架构。
2. 在顶部的中间面板中，对于无法自动转换为目标数据库引擎的源数据库引擎，将显示针对其中架构元素的操作项。

3. 在右侧面板中，目标数据库实例中的架构会显示在树状图中。您的数据库架构会“延迟加载”。换而言之，当您从树状图中选择某个项时，AWS SCT 会从您的目标数据库中获取并显示当前架构。



4. 选择架构元素时，左下方的面板中将显示属性。它们描述了源架构元素和在源数据库中创建该元素的 SQL 命令。
5. 选择架构元素时，右下方的面板中将显示属性。它们描述了目标架构元素和在目标数据库中创建该元素的 SQL 命令。您可以编辑此 SQL 命令并将更新的命令连同您的项目一起保存。

启动 AWS SCT

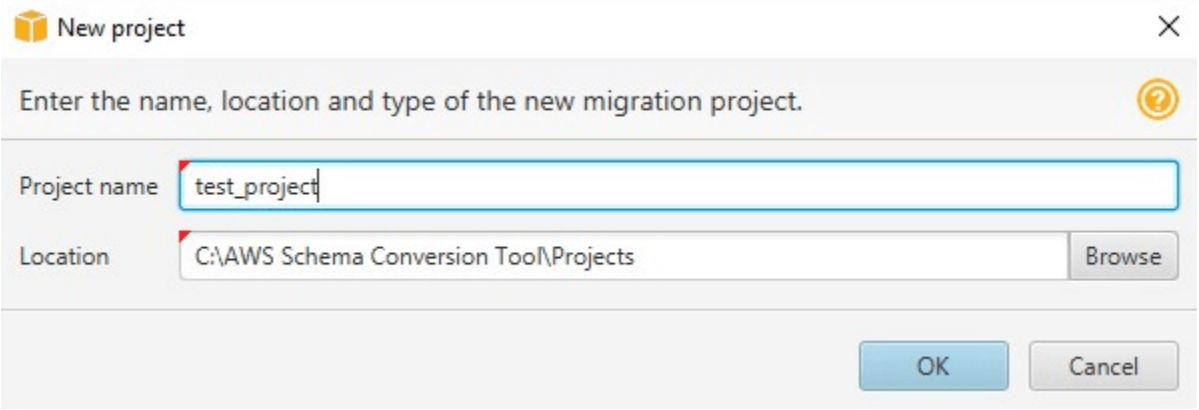
要启动 AWS Schema Conversion Tool，双击该应用程序的图标。

创建 AWS SCT 项目

使用以下过程创建 AWS Schema Conversion Tool 任务。

创建您的项目

1. 启动 AWS Schema Conversion Tool。
2. 在文件菜单上，选择新建项目。此时将显示新建项目对话框。



3. 为本地存储在计算机上的项目输入一个名称。
4. 输入本地项目文件的位置。
5. 选择 OK (确定) 以创建您的 AWS SCT 项目。
6. 选择添加源，将新的源数据库添加到 AWS SCT 项目中。您可以向 AWS SCT 项目中添加多个源数据库。
7. 选择添加目标可在 AWS SCT 项目中添加新的目标平台。您可以向 AWS SCT 项目添加多个目标平台。
8. 在左侧面板中选择源数据库架构。
9. 在右侧面板中，为所选源架构指定目标数据库平台。
10. 选择创建映射。选择源数据库架构和目标数据库平台后，此按钮将变为活动状态。有关更多信息，请参阅创建映射规则。

现在，AWS SCT 项目已设置完毕。您可以保存项目、创建数据库迁移评估报告和转换源数据库架构。

在 AWS SCT 中使用新的项目向导

您可以使用新建项目向导创建新的数据库迁移项目。此向导可帮助您确定迁移目标并连接到数据库。它可以估计所有支持的目标目的地的迁移的复杂程度。运行向导后，AWS SCT 将生成数据库迁移到不同目标目的地的摘要报告。您可以使用此报告比较可能的目标目的地并选择最佳迁移路径。

运行新项目向导

1. 选择数据源。

   a. 启动 AWS Schema Conversion Tool。

   b. 在文件菜单上，选择新建项目向导。将打开创建新的数据库迁移项目对话框。

   c. 要输入源数据库连接信息，请按照以下说明进行操作：

参数	操作
项目名称	为本地存储在计算机上的项目输入一个名称。
位置	输入本地项目文件的位置。
Source type（源类型）	选择以下选项之一：SQL 数据库、NoSQL 数据库或 ETL。 如果要查看包含所有迁移目的地的摘要报告，请选择 SQL 数据库。
源引擎	选择源数据库引擎。
迁移策略	请选择以下任一选项： • 我想切换引擎并优化云：此选项可将源数据库转换为新的数据库引擎。 • 我想保留相同的引擎，但要优化云：此选项可保持数据库引擎不变，并将数据库从本地迁移到云端。 • 我想查看数据库引擎切换和云优化的合并报告：此选项比较了所有可用迁移选项的迁移复杂性。 如果您想查看包含所有迁移目的地的汇总评估报告，请选择最后一个选项。

   d. 选择下一步。此时将打开连接到源数据库页面。

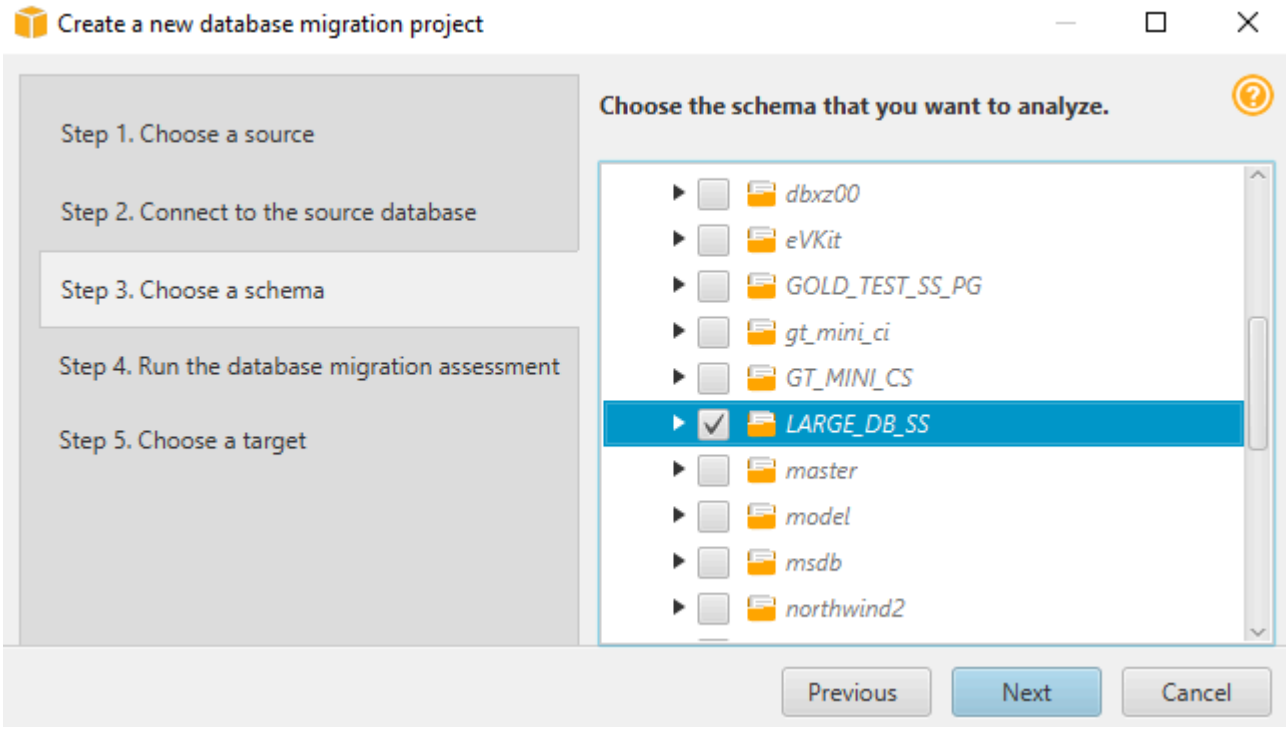
2. 连接到您的源数据库。

   a. 提供源数据库的连接信息。连接参数取决于源数据库引擎。确保用于分析源数据库的用户具有适用的权限。有关更多信息，请参阅AWS SCT 的源。

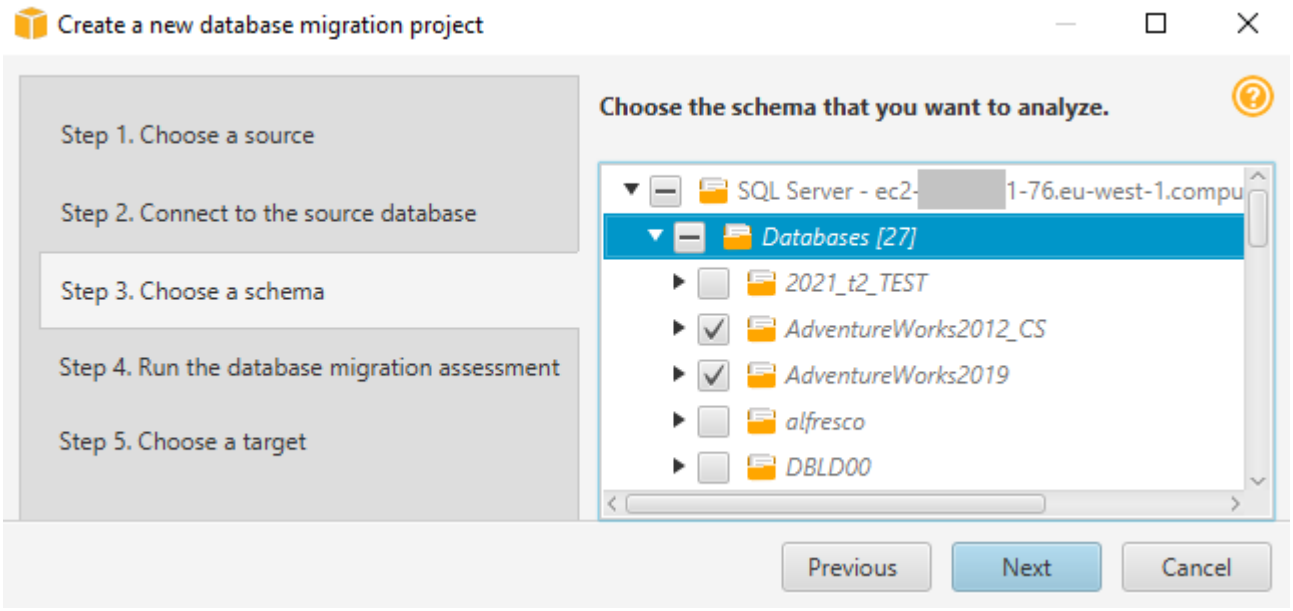
   b. 选择下一步。此时将打开选择架构页面。

3. 选择数据库架构。

   a. 选中要评估的架构名称对应的复选框，然后选择架构。选择架构名称后，它将以蓝色突出显示，且下一步按钮可用。



   b. 如果要评估多个数据库架构，请选中所有架构的复选框，然后选择父节点。要成功进行评估，必须选择父节点。例如，对于源 SQL Server 数据库，请选择数据库节点。父节点的名称以蓝色突出显示，且下一步按钮可用。



c. 选择下一步。AWS SCT 分析源数据库架构并创建数据库迁移评估报告。源数据库架构中数据库对象的数量会影响评估运行所需的时间。完成后，将打开运行数据库迁移评估页面。

4. 运行数据库迁移评估。

   a. 您可以查看和比较不同迁移目标的评估报告，也可以保存评估报告文件的本地副本以供进一步分析。

   b. 保存数据库迁移评估报告的本地副本。选择保存，然后输入保存文件的文件夹路径，然后选择保存。AWS SCT 将评估报告文件保存到指定文件夹。

   c. 选择下一步。此时将打开选择目标页面。

5. 选择目标数据库。

   a. 对于目标引擎，请根据评估报告选择您决定使用的目标数据库引擎。

   b. 提供目标数据库连接信息。您看到的连接参数取决于所选的目标数据库引擎。确保为目标数据库指定的用户具有所需的权限。有关所需权限的更多信息，请参阅 AWS SCT 的源 和 将 Amazon Redshift 作为目标的权限 中描述目标数据库权限的部分。

   c. 选择完成。AWS SCT 创建项目并添加映射规则。有关更多信息，请参阅创建映射规则。

现在，您可以使用 AWS SCT 项目转换源数据库对象。

保存和打开 AWS SCT 项目

使用以下过程保存 AWS Schema Conversion Tool 项目。

保存项目

1. 启动 AWS Schema Conversion Tool。
2. 在文件菜单上，选择保存项目。

   AWS SCT 将项目保存到您在创建项目时指定的文件夹中。

使用以下过程打开现有 AWS Schema Conversion Tool 项目。

打开项目

1. 从文件菜单中，选择打开项目。随即显示打开对话框。
2. 选择项目文件夹，然后选择 Windows 脚本组件（*.sct）文件。
3. AWS SCT 打开项目，但不会自动连接到源数据库和目标数据库。选择数据库架构树顶部的连接到服务器，连接至源数据库和目标数据库。

如果打开保存在 AWS SCT 版本 1.0.655 或更早版本中的项目，则 AWS SCT 会自动为所有源数据库架构创建到目标数据库平台的映射规则。要添加其他目标数据库平台，请删除现有的映射规则，然后创建新的映射规则。有关创建映射规则的更多信息，请参阅 创建映射规则。

向 AWS SCT 项目添加数据库服务器

您可以将多个源数据库服务器和目标数据库服务器添加到一个 AWS Schema Conversion Tool 项目中。

将服务器添加到项目。

1. 启动 AWS Schema Conversion Tool。
2. 创建新项目，或打开现有项目。
3. 从菜单中选择添加源以添加新的源数据库。
4. 选择数据库平台并指定数据库连接凭证。有关连接到源数据库的更多信息，请参阅 AWS SCT 的源。

按照以下过程连接到数据库。

连接到数据库。

1. 打开数据库服务器的上下文（右键单击）菜单，然后选择建立连接。

   您也可以在数据库架构树的顶部选择连接到服务器。
2. 输入用于连接到源数据库服务器的密码。
3. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
4. 选择连接以连接到源数据库。

使用以下过程从 AWS SCT 项目中移除数据库服务器。

移除数据库服务器

1. 选择要移除的数据库服务器。
2. 打开上下文菜单（单击右键），然后选择从项目中移除。

   AWS SCT 移除选定的数据库服务器、所有映射规则、转换结果以及与此服务器相关的其他元数据。

在离线模式下运行 AWS SCT

你可以在离线模式下运行 AWS Schema Conversion Tool。接下来，您可以了解如何在与源数据库断开连接时使用现有 AWS SCT 项目。

AWS SCT 不需要连接到源数据库即可运行以下操作：

- 添加映射规则。
- 创建数据库迁移评估报告。
- 转换数据库架构和代码。
- 编辑源代码和转换后的代码。
- 将源代码和转换后的代码作为 SQL 脚本保存到文本文件中。

在离线模式下使用 AWS SCT 之前，请先连接到源数据库，加载元数据并保存项目。打开此项目或断开与源数据库服务器的连接，以便在脱机模式下使用 AWS SCT。

在离线模式下运行 AWS SCT

1. 启动 AWS Schema Conversion Tool 并创建一个新项目。有关更多信息，请参阅创建 AWS SCT 项目。
2. 添加源数据库服务器并连接到源数据库。有关更多信息，请参阅向 AWS SCT 项目添加数据库服务器。
3. 添加目标数据库服务器或使用虚拟目标数据库平台。有关更多信息，请参阅使用虚拟目标。
4. 创建映射规则以定义源数据库的目标数据库平台。有关更多信息，请参阅在 AWS SCT 中创建映射规则。
5. 选择视图，然后选择主视图。
6. 在显示源数据库对象的左侧面板中，选择源数据库架构。打开该对象的上下文 (右键单击) 菜单，然后选择加载架构。此操作将所有源架构元数据加载到 AWS SCT 项目中。

   创建报告和转换架构操作还会将所有源架构元数据加载到 AWS SCT 项目中。如果从上下文菜单运行其中一个操作，请跳过加载架构操作。
7. 在文件菜单上，选择保存项目，将源数据库元数据保存在项目中。
8. 选择断开与服务器的连接，断开与源数据库的连接。现在，您可以在离线模式下使用 AWS SCT。

使用 AWS SCT 树筛选器

为了将数据从源迁移到目标，AWS SCT 会将所有元数据从源数据库和目标数据库加载到一个树结构中。此结构作为主项目窗口中的树视图显示在 AWS SCT 中。

某些数据库可能有大量对象在树结构中。您可以使用 AWS SCT 中的树筛选器在源和目标树结构中搜索对象。使用树筛选器时，您无需更改转换数据库时已转换过的对象。筛选器仅更改树状图中可以看到的内容。

树筛选器可与 AWS SCT 已预加载的对象结合使用。换而言之，AWS SCT 在搜索期间不会从数据库加载对象。此方法意味着树结构包含的对象通常比数据库中存在的更少。

对于树筛选器，请注意以下事项：

- 筛选器默认值为 ANY，这意味着筛选器使用名称搜索查找对象。
- 当您选择一个或多个对象类型时，您只会在树中看到这些类型的对象。
- 您可以使用筛选器掩码显示不同类型的符号，包括 Unicode、空格和特殊字符。“%”字符是任何符号的通配符。
- 您应用筛选器后，计数将仅显示已筛选对象数。

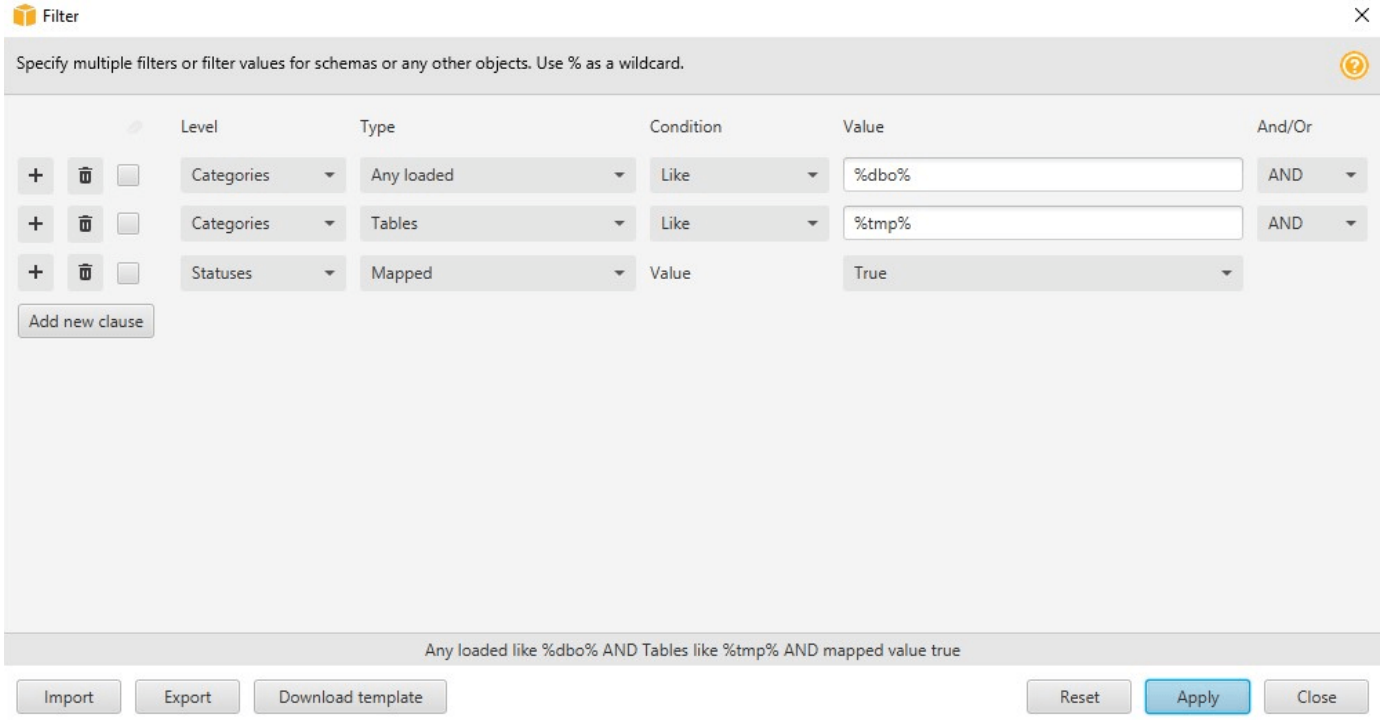
创建树筛选器

1. 打开现有 AWS SCT 项目。
2. 连接到您要应用树筛选器的数据库。
3. 选择“筛选器”图标。

“撤消筛选器”图标将灰显，因为目前没有应用筛选器。

4. 在筛选条件对话框中输入以下信息。对于每个数据库引擎，该对话框中的选项都不同。

AWS SCT 筛选条件选项	操作
级别	选择类别可按类别筛选对象。 选择状态可按状态筛选对象。
Type	对于级别中的类别，请选择已筛选对象的类别。选择任意已加载以显示所有类别中的对象。 对于级别中的状态，请选择已筛选对象的状态。您可以选择以下选项之一： • 已转换显示所有已转换的对象 • 需操作显示所有存在转换问题的对象 • 已加密显示所有加密对象
Condition	对于级别中的类别，在喜欢和不喜欢之间选择筛选条件。 对于级别中的状态，筛选条件选项不可用。
Value	对于级别中的类别，请输入值以按此值筛选树。 使用百分比 (%) 作为通配符来显示所有对象。 对于级别中的状态，请选择介于 True 和 False 之间的值。
和/或	选择 AND 或 OR 逻辑运算符以应用多个筛选子句。



5. 选择添加新子句以添加其他筛选子句。AWS SCT 可以使用 AND 或 OR 逻辑运算符应用多个筛选子句。
6. 选择应用。选择 Apply 后，“撤消筛选器”图标 (在“筛选器”图标旁边) 就会启用。如果要删除您应用的筛选器，请使用此图标。
7. 选择 Close 以关闭此对话框。

当您筛选显示在树状图中的架构时，无需更改转换架构时已转换过的对象。筛选器仅更改树状图中可以看到的内容。

为树筛选器导入文件列表

您可以导入带有分号分隔符的逗号分隔值（CSV）文件，也可以导入包含您希望树过滤器使用的名称或值的 JSON 文件。打开现有 AWS SCT 项目，连接到要应用树筛选器的数据库，然后选择“筛选器”图标。

要下载文件示例，请选择下载模板。输入文件名称，然后选择保存。

要下载现有的筛选器设置，请选择导出。输入文件名称，然后选择保存。

要为树筛选器导入文件列表，请选择导入。选择要导入的文件，然后选择 Open。选择 Apply，然后选择 Close。

CSV 文件将分号作为分隔符，格式如下：

- object_type 是您要查找的对象的类型。
- database_name 是其中存在此对象的数据库的名称。
- schema_name 是其中存在此对象的架构的名称。
- object_name 是对象名称。
- import_type 指定从筛选器中 include 或 exclude 此项目。

使用 JSON 文件描述复杂的筛选案例，例如嵌套规则。JSON 文件具有如下格式：

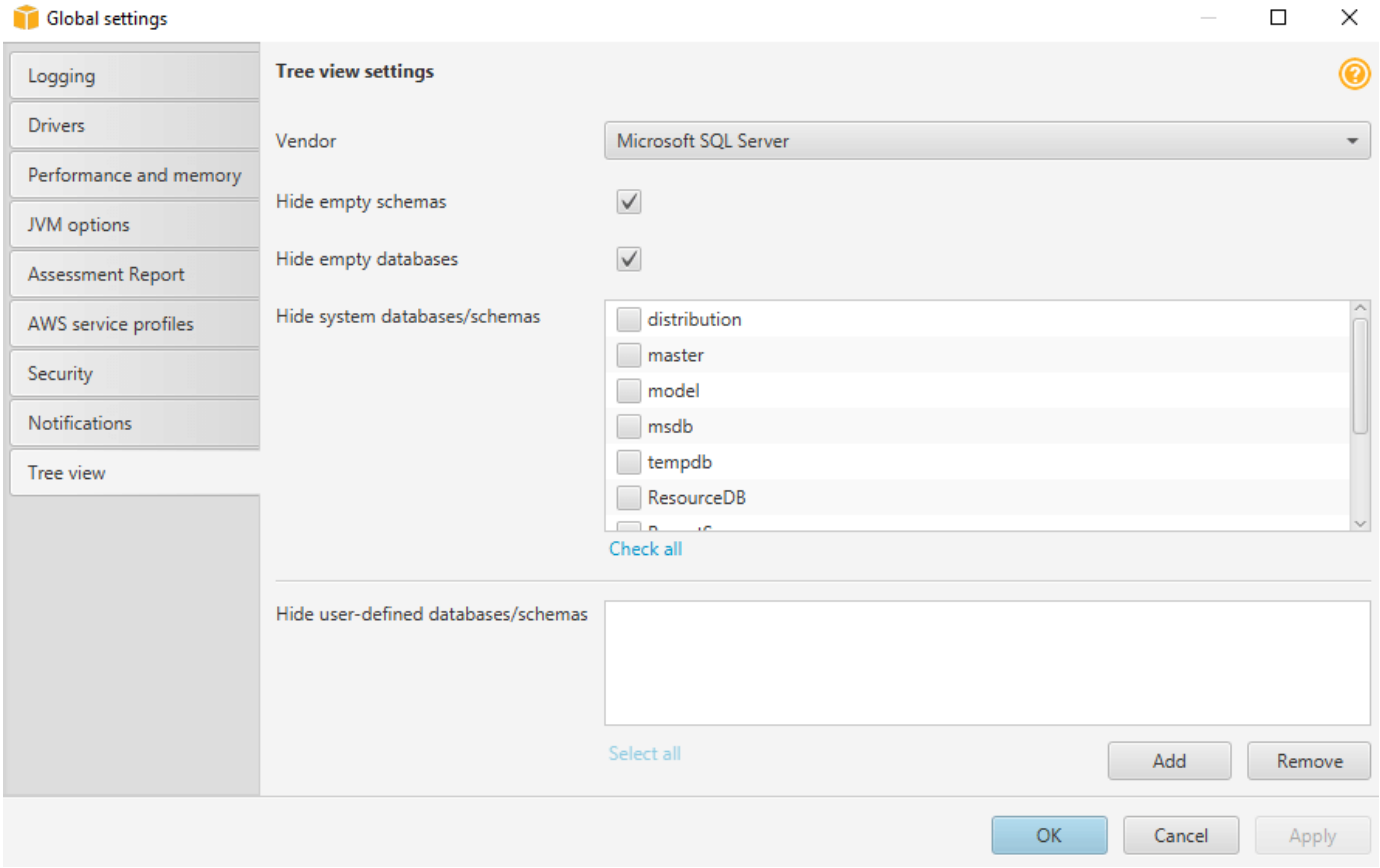
- filterGroupType 是应用于多个筛选条件子句的筛选规则（AND 或 OR 逻辑运算符）的类型。
- filterCategory 是筛选器的级别（类别或状态）。
- names 是适用于类别筛选器的对象名称列表。
- filterCondition 是适用于类别筛选器的筛选条件（LIKE 或 NOT LIKE）。
- transformName 是适用于状态筛选器的状态名称。
- value 是筛选树所依据的值。
- transformValue 是适用于状态筛选器的筛选器（TRUE 或 FALSE）的值。

在 AWS SCT 树视图中隐藏架构

利用树视图设置，您可以指定要在 AWS SCT 树视图中查看的架构和数据库。您可以隐藏空架构、空数据库、系统数据库以及用户定义的数据库和架构。

在树视图中隐藏数据库和架构

1. 打开 AWS SCT 项目。
2. 连接到要在树视图中显示的数据存储。
3. 依次选择设置、全局设置、树视图。



4. 在树视图设置部分，执行以下操作：

   - 对于供应商，选择数据库平台。
   - 选择隐藏空架构可隐藏所选数据库平台的空架构。
   - 选择隐藏空数据库以隐藏所选数据库平台的空数据库。
   - 对于隐藏系统数据库/架构，按名称选择系统数据库和架构以隐藏它们。
   - 对于隐藏用户定义的数据库/架构，输入要隐藏的用户定义的架构和数据库的名称，然后选择添加。名称不区分大小写。

5. 选择确定。

创建和查看数据库迁移评估报告

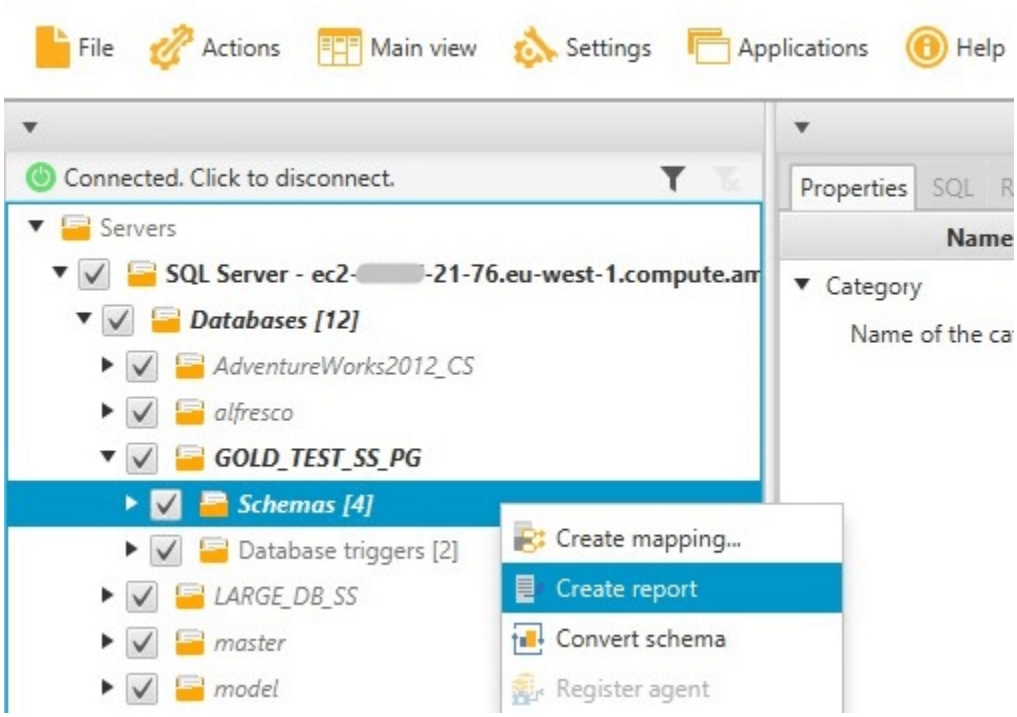
数据库迁移评估报告总结了无法自动转换为目标 Amazon RDS 数据库实例引擎的架构的所有操作项。此外，该报告还包含为目标数据库实例编写等效代码所需工作量的估算。

在将源数据库和目标平台添加到项目中并指定映射规则之后，您可以创建数据库迁移评估报告。

创建和查看数据库迁移评估报告

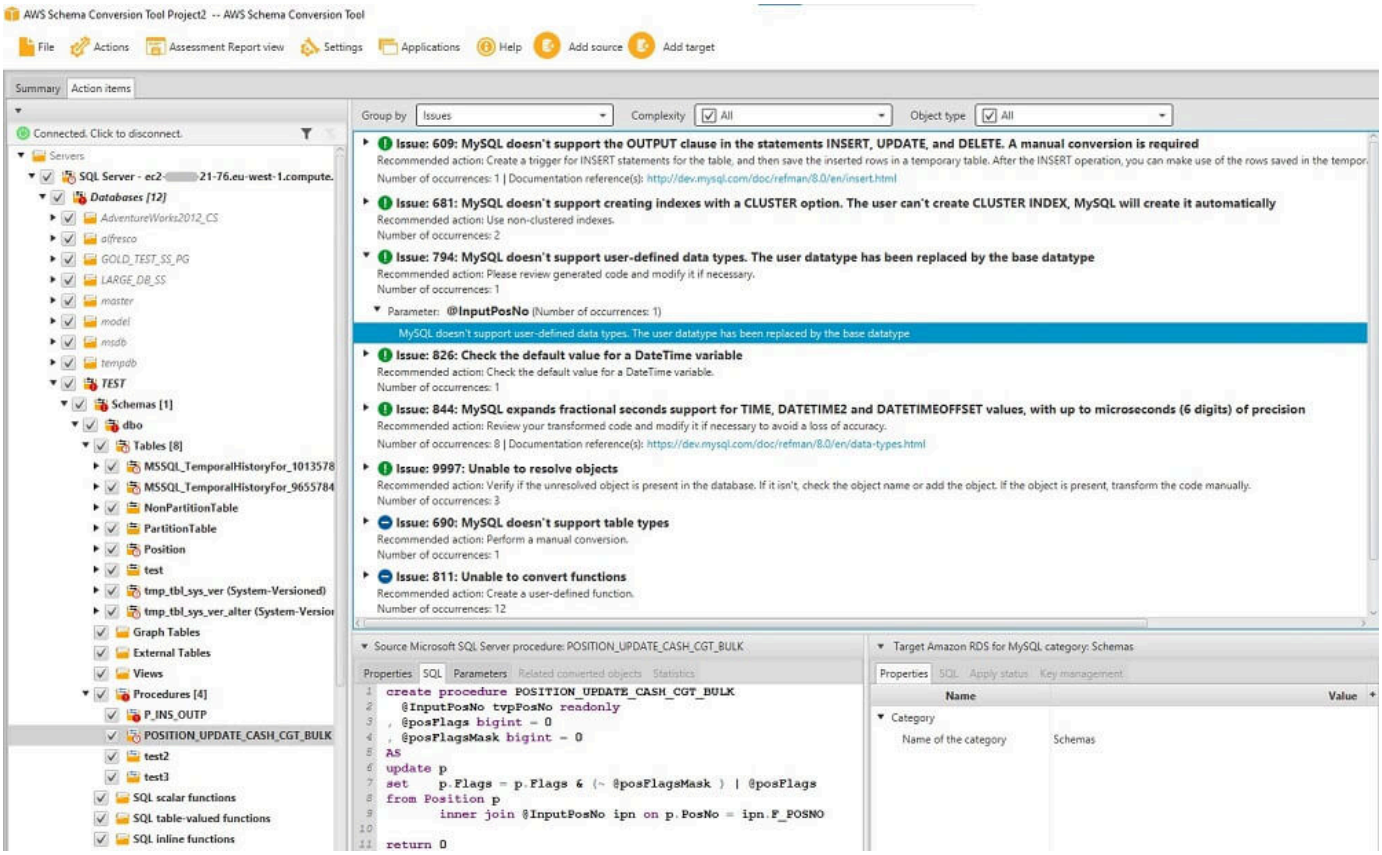
1. 确保您创建了源数据库架构映射规则，以便为其创建评估报告。有关更多信息，请参阅添加新映射规则。
2. 在视图菜单上，选择主视图。
3. 在显示源数据库架构的左侧面板中，选择要创建评估报告的架构对象。

   确保选中了要创建评估报告的所有架构对象的复选框。
4. 打开该对象的上下文 (右键单击) 菜单，然后选择创建报告。



   评估报告视图会打开。
5. 选择操作项选项卡。

   操作项选项卡显示描述无法自动转换的架构的项目列表。从列表中选择其中一个操作项。AWS SCT 突出显示架构中该操作项适用的项，如下所示。

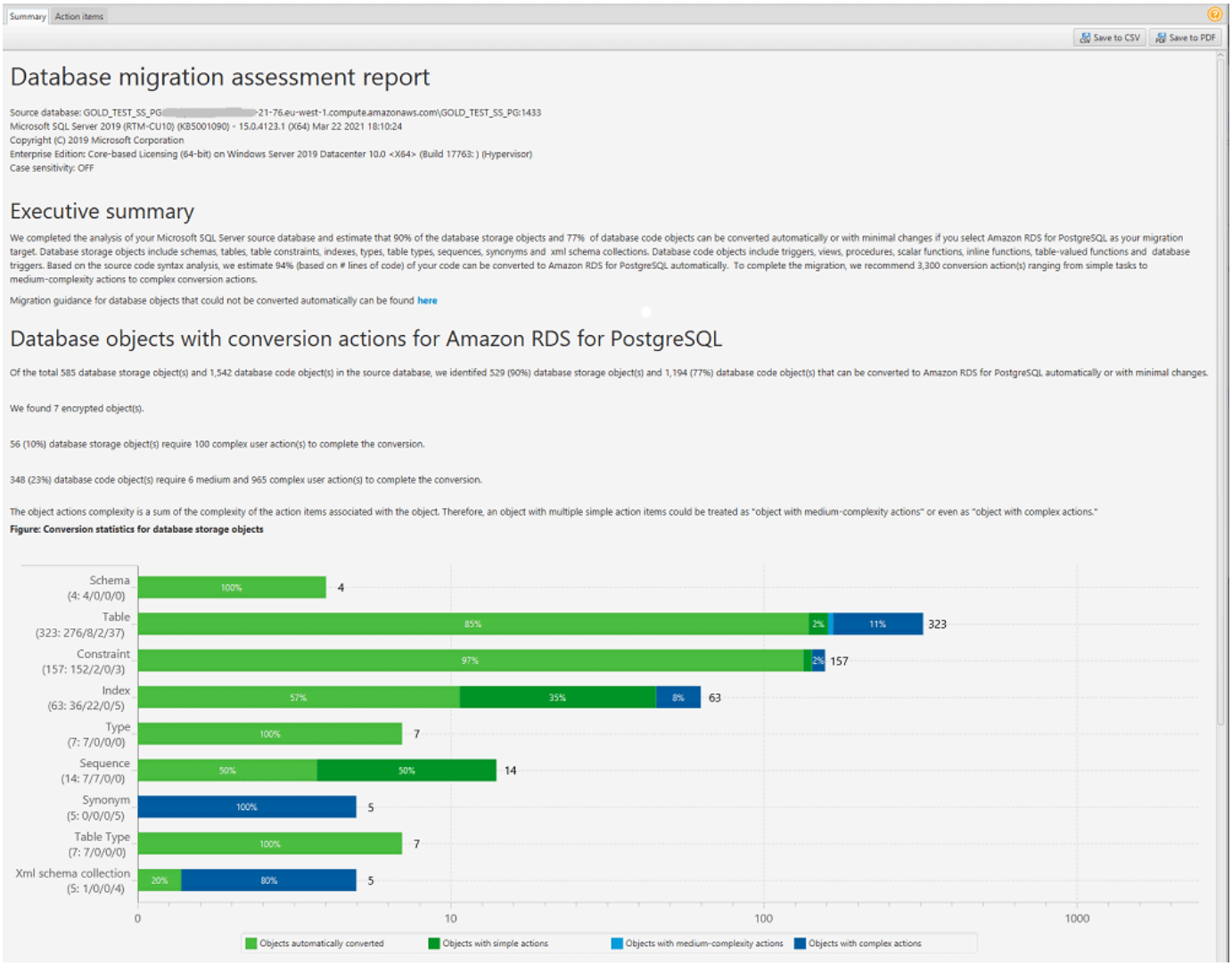


6. 选择 Summary 选项卡。

   Summary 选项卡显示了来自数据库迁移评估报告的摘要信息。它显示已自动转换的项目数量和未自动转换的项目数量。该摘要还包含在目标数据库实例中创建与源数据库中架构等效的架构所需的时间的估算。

   许可证评估和云支持部分包含关于将现有的本地数据库架构移到运行相同引擎的 Amazon RDS 数据库实例的信息。例如，如果您希望更改许可证类型，报告的此部分将告诉您应从当前数据库中删除哪些功能。

   评估报告摘要的示例如下所示。



7. 选择 Summary 选项卡，然后选择 Save to PDF。数据库迁移评估报告会被另存为 PDF 文件。PDF 文件包含摘要和操作项信息。

   您也可以选择保存为 CSV 将报告另存为逗号分隔值（CSV）文件。选择此选项后，AWS SCT 会创建三个 CSV 文件。这些文件包含以下信息：

   - 包含推荐操作的转化操作项列表。
   - 转换操作项的摘要，包括转换操作项的发生次数所需的估计工作量。
   - 一份执行摘要，其中包含许多按预计转换时间分类的操作项。

Database objects with conversion actions for Amazon RDS for PostgreSQL

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

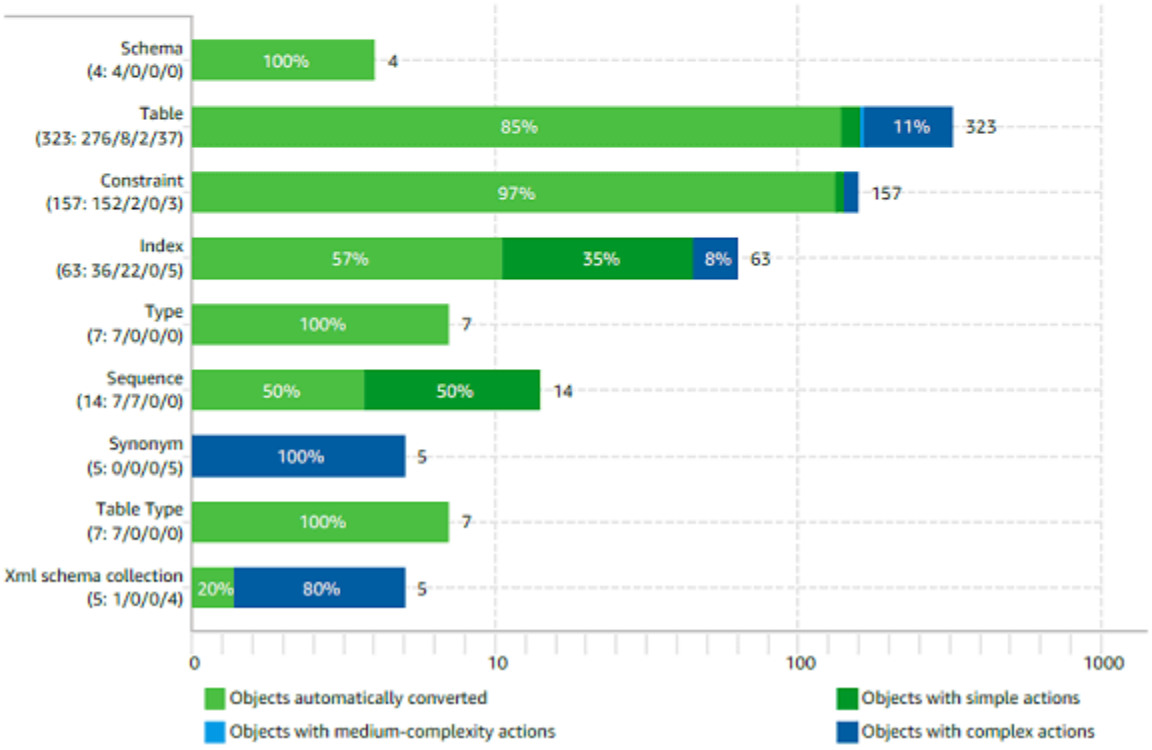
We found 7 encrypted object(s).

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

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

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

Figure: Conversion statistics for database storage objects

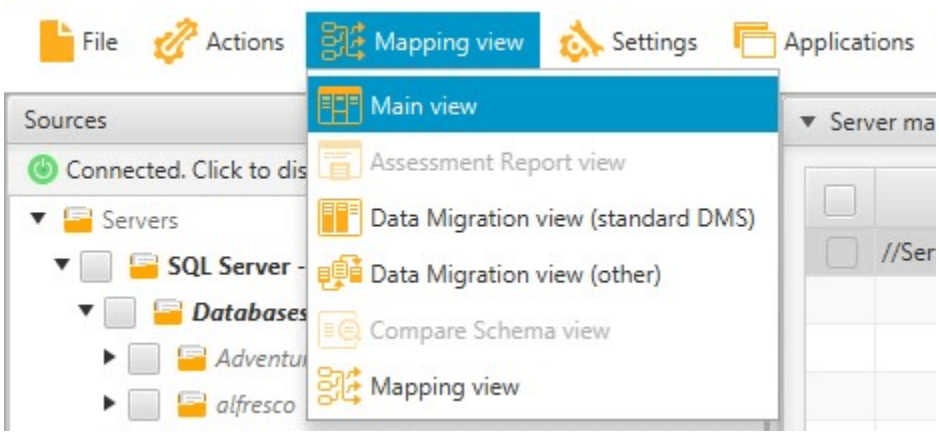


转换架构

在将源数据库和目标数据库添加到项目中并创建映射规则后，您可以转换源数据库架构。使用以下过程转换架构。

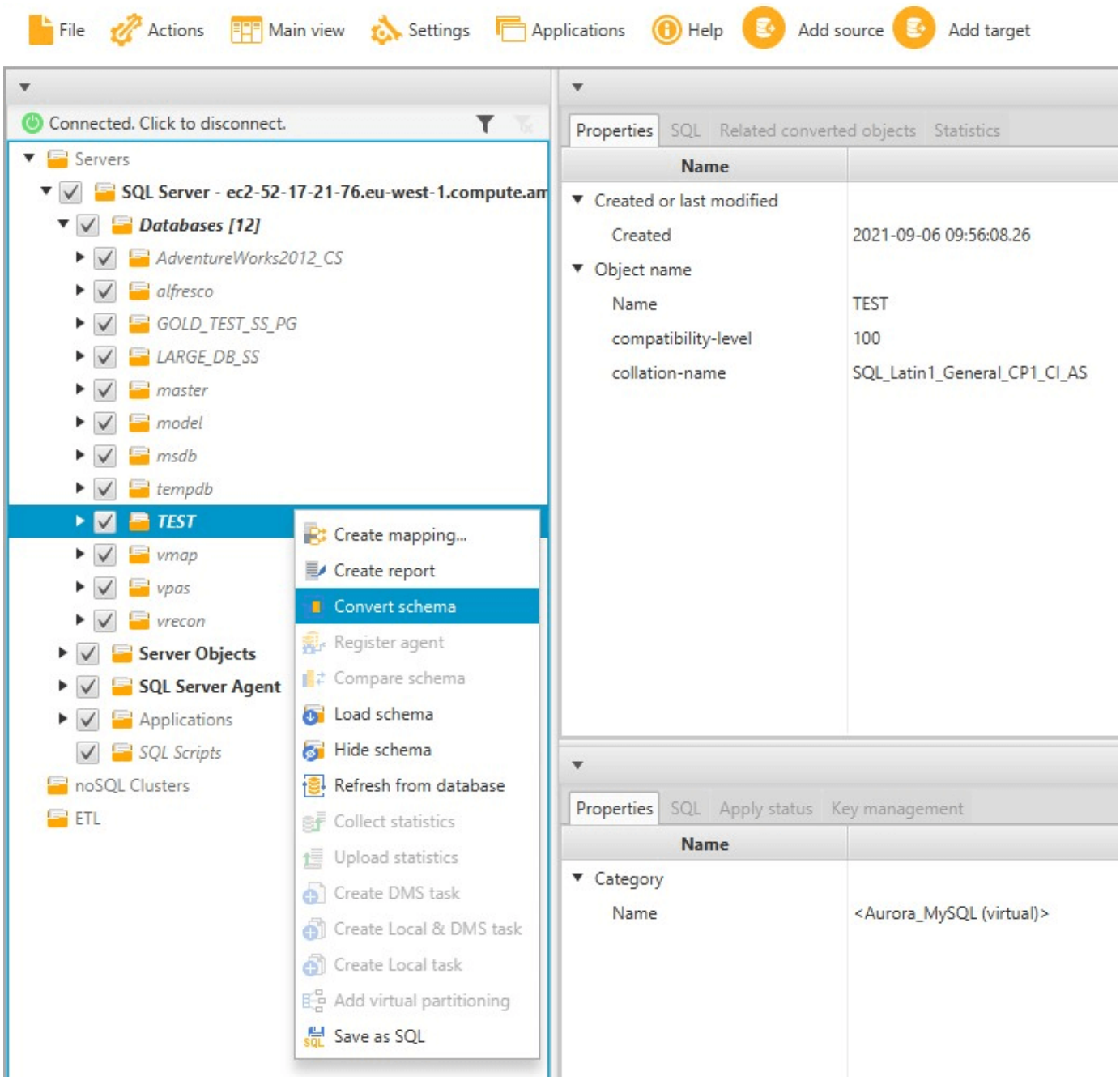
转换架构

1. 选择视图，然后选择主视图。



2. 在显示源数据库架构的左侧面板中，选择要转换的对象名称对应的复选框。接下来，选择此对象。AWS SCT 用蓝色突出显示对象名称。打开该对象的上下文 (右键单击) 菜单，然后选择转换架构。

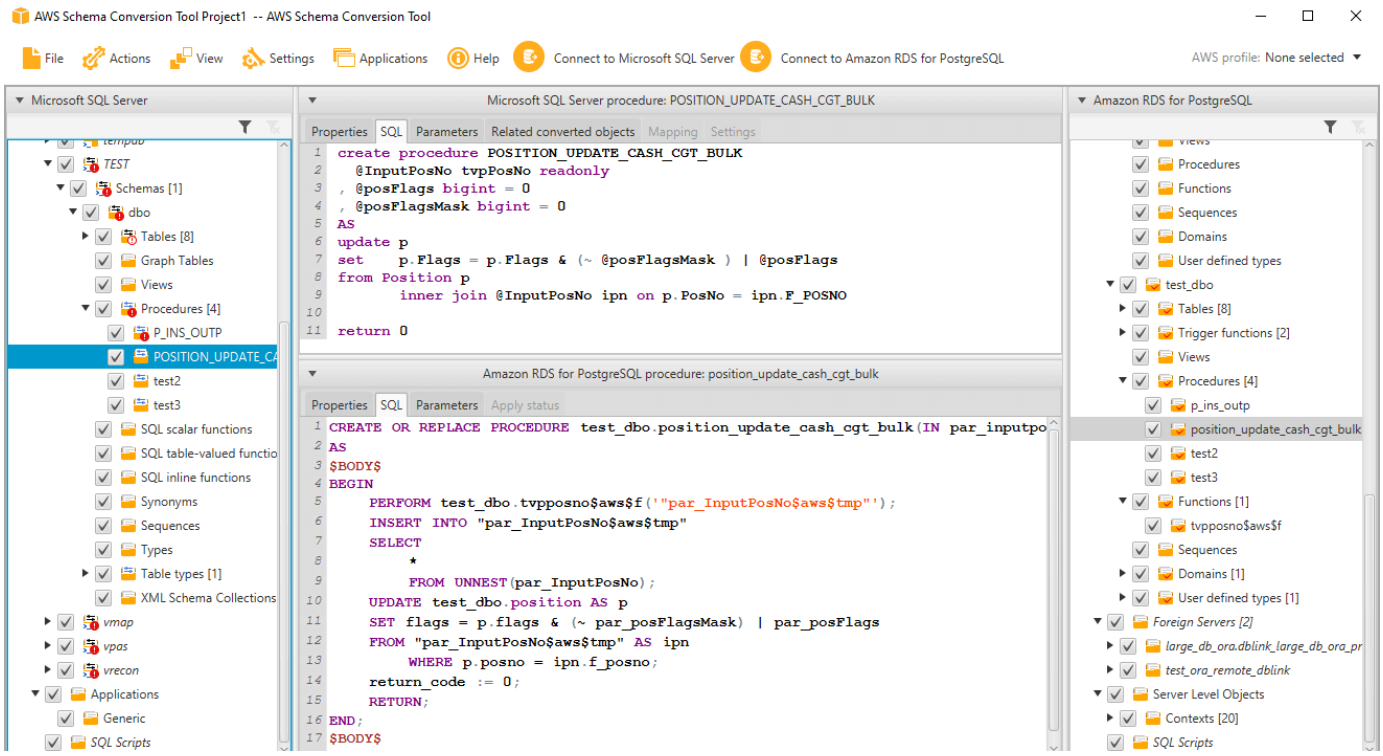
   要转换多个数据库对象，请选中所有对象对应的复选框。接下来，选择父节点。例如，对于表，父节点是表。确保 AWS SCT 以蓝色突出显示父节点的名称。打开该父节点的上下文 (右键单击) 菜单，然后选择转换架构。



3. 当 AWS SCT 完成架构转换时，您可以在项目右侧面板中查看提议的架构。

   此时，没有任何架构应用于目标数据库实例。计划的架构是您项目的一部分。如果选择一个转换后的架构项，则可在中下方面板中看到目标数据库实例的计划架构命令。

   您可以在此窗口中编辑架构。在您选择应用转换后的架构时，编辑后的架构会作为项目的一部分进行存储，并写入目标数据库实例。



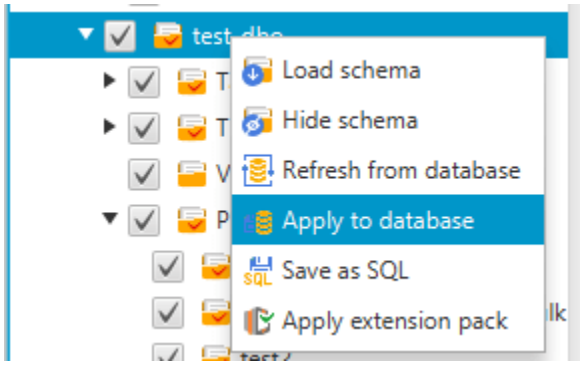
将转换后的架构应用于目标数据库实例

您可以将转换后的数据库架构应用于目标数据库实例。当该架构已应用于目标数据库实例后，您可以根据数据库迁移评估报告中的操作项来更新该架构。

> **Warning**
>
> 以下过程将覆盖现有的目标架构。请务必小心，不要意外覆盖架构。请务必小心，不要覆盖已修改的目标数据库实例中的架构，否则将覆盖这些更改。

将转换后的数据库架构应用于目标数据库实例

1. 选择项目右侧面板顶部的连接到服务器以连接到目标数据库。如果已连接到目标数据库，请跳过此步骤。
2. 在显示目标数据库实例的计划架构的项目右侧面板中选择架构元素。
3. 打开架构元素的上下文 (右键单击) 菜单，然后选择 Apply to database。



转换后的架构将应用于目标数据库实例。

将 AWS 服务配置文件存储在 AWS SCT 中

您可以将 AWS 凭证存储在 AWS SCT 中。当您使用与 AWS 服务集成的功能时，AWS SCT 会使用该凭证。例如，AWS SCT 与 Amazon S3、AWS Lambda、Amazon Relational Database Service（Amazon RDS）和 AWS Database Migration Service（AWS DMS）集成。

当您访问需要 AWS 凭证的功能时，AWS SCT 会要求您提供凭证。您可以将凭证存储在全局应用程序设置中。当 AWS SCT 要求您提供凭证时，您可以选择存储的凭证。

您可以将不同的 AWS 凭证组存储在全局应用程序设置中。例如，您可以存储一组在测试场景中使用的凭证，并存储另一组在生产场景中使用的凭证。您也可以针对不同的 AWS 区域 区域存储不同的凭证。

存储 AWS 凭证

使用以下过程可全局存储 AWS 凭证。

存储 AWS 凭证

1. 启动 AWS Schema Conversion Tool。
2. 打开设置菜单，然后选择全局设置。此时显示 Global settings 对话框。
3. 选择 AWS 服务配置文件，然后选择添加新的 AWS 服务配置文件。
4. 按照以下所示输入 AWS 信息。

AWS SCT 选项	操作
配置文件名称	输入配置文件的名称。

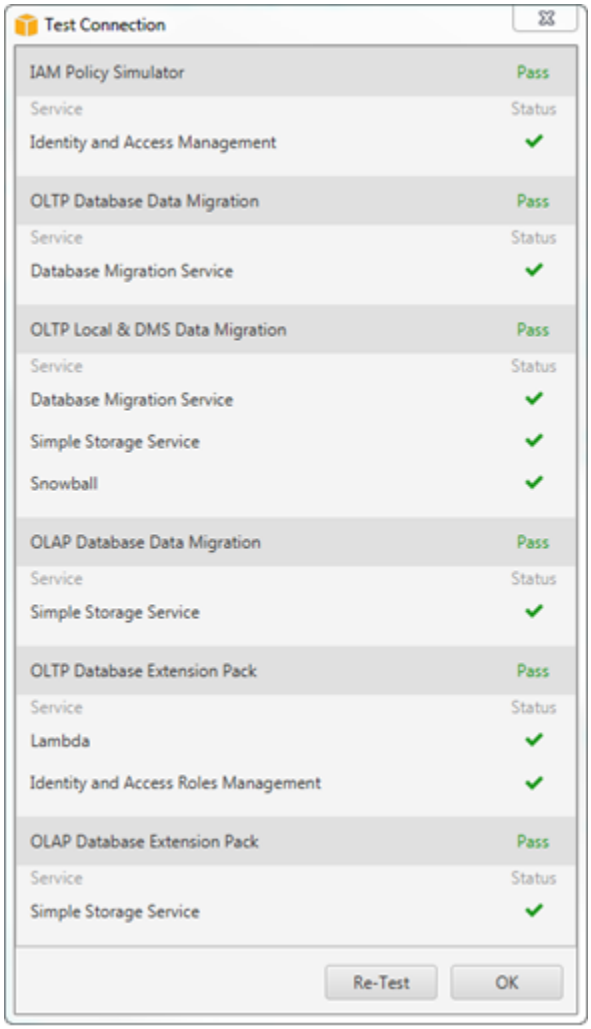
AWS SCT 选项	操作
AWS 访问密钥	输入 AWS 访问密钥。
AWS 私有密钥	输入 AWS 秘密访问密钥。有关 AWS 访问密钥的更多信息，请参阅《IAM 用户指南》中的管理访问密钥。
区域	为配置文件选择 AWS 区域。
Amazon S3 存储桶文件夹	请为配置文件选择 Amazon S3 存储桶。只有在使用连接到 Amazon S3 的功能时，才需要指定存储桶。有关所需权限的更多信息，请参阅使用 AWS 服务配置文件的权限。

如果您需要符合联邦信息处理标准 (FIPS) 的安全要求，请选择 Use FIPS endpoint for S3。FIPS 端点在以下 AWS 区域可用：

- 美国东部（弗吉尼亚州北部）区域
- 美国东部（俄亥俄州）区域
- 美国西部（加利福尼亚北部）区域
- 美国西部（俄勒冈州）区域

5. 选择测试连接以验证凭证是否正确以及是否有效。

   此时显示测试连接对话框。您可以查看每个连接到配置文件的服务的状态。Pass 表示配置文件可成功访问该服务。



6. 配置完配置文件后，请选择 Save 以保存您的配置文件，或选择 Cancel 以取消您的更改。
7. 选择确定以关闭全局设置对话框。

设置项目的默认配置文件

您可以设置 AWS SCT 项目的默认配置文件。此操作可将配置文件中存储的 AWS 凭证与项目进行关联。打开项目，使用以下过程设置默认配置文件。

设置项目的默认配置文件

1. 启动 AWS Schema Conversion Tool 并创建一个新项目。
2. 在设置菜单上，选择项目设置。随即出现项目设置对话框。
3. 选择项目环境选项卡。

4. 选择添加新的 AWS 服务配置文件以添加新的配置文件。对于 AWS 服务配置文件，请选择您要与项目关联的配置文件。
5. 选择确定以关闭项目设置对话框。您也可以选择 Cancel 取消所做更改。

使用 AWS 服务配置文件的权限

从 AWS 服务配置文件访问 Amazon S3 存储桶需要以下权限：

- `s3:PutObject`：在 Amazon S3 存储桶中添加对象。
- `s3:DeleteObject`：删除对象的空版本并插入删除标记，此版本成为对象的当前版本。
- `s3:ListBucket`：从 Amazon S3 存储桶中返回最多 1000 个对象。
- `s3:GetObject`：从 Amazon S3 存储桶中检索对象。

以下代码示例显示如何将这些权限授予您的用户。

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

使用 AWS Secrets Manager

AWS SCT 可以使用您存储在 AWS Secrets Manager 中的数据库凭证。您可以在 Secrets Manager 的数据库连接对话框中填写所有值。要使用 Secrets Manager，请确保将 AWS 配置文件存储在 AWS Schema Conversion Tool 中。

有关使用 AWS Secrets Manager 的更多信息，请参阅《AWS Secrets Manager 用户指南》中的什么是 AWS Secrets Manager？。有关存储 AWS 配置文件的更多信息，请参阅 将 AWS 服务配置文件存储在 AWS SCT 中。

从 Secrets Manager 中检索数据库凭证

1. 启动 AWS Schema Conversion Tool 并创建一个新项目。
2. 选择添加源或添加目标，将新数据库添加到项目中。
3. 选择数据库平台，然后选择下一步。
4. 对于 AWS 密钥，选择要使用的密钥。
5. 选择填充。然后 AWS SCT 在数据库连接对话框中填写所有值。
6. 选择测试连接以验证 AWS SCT 是否可以连接到数据库。
7. 选择连接以连接到您的数据库。

AWS SCT 支持具有如下结构的密钥。

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

在此结构中，`username` 和 `password` 值是必填的，所有其他值都是可选的。确保您存储在 Secrets Manager 中的值包含所有数据库凭证。

存储数据库密码

您可以将数据库密码或 SSL 证书存储在 AWS SCT 缓存中。要存储密码，请在创建连接时选择 Store Password (存储密码)。

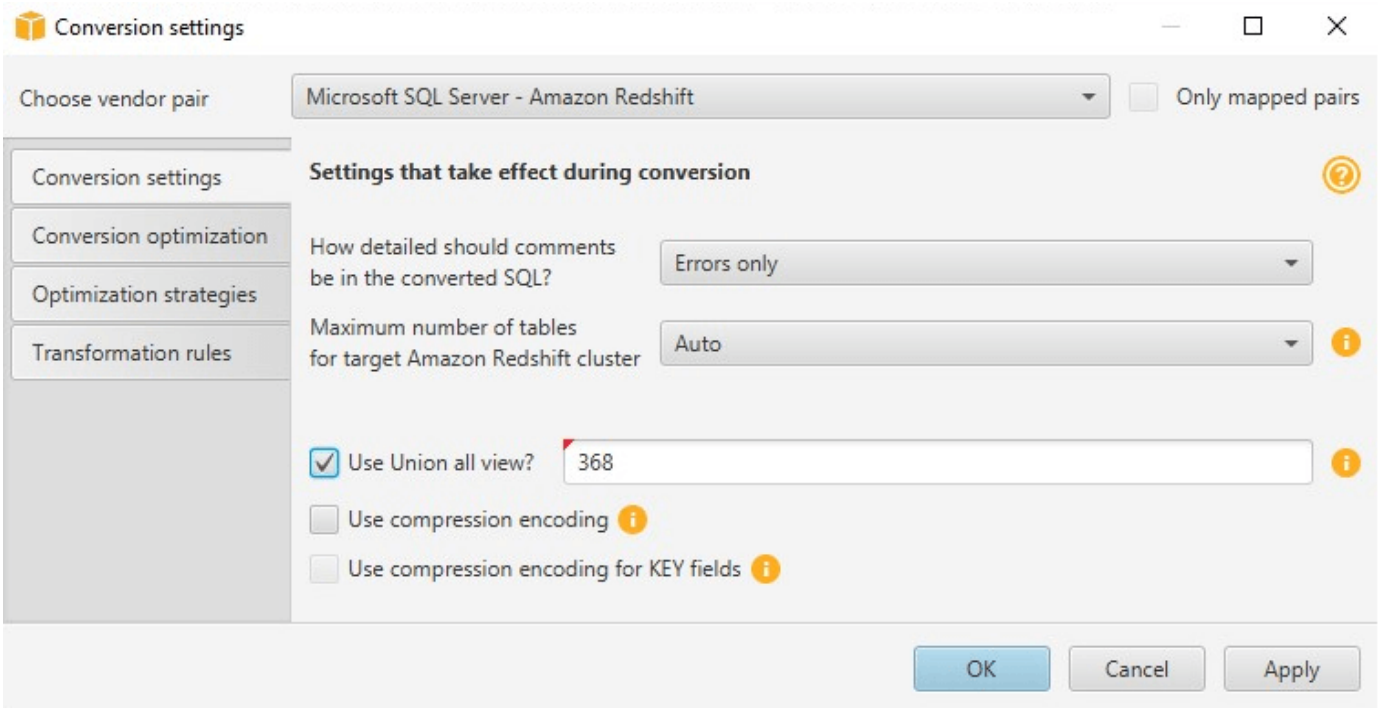
您可使用 `seed.dat` 文件中随机生成的令牌对密码进行加密，然后使用用户名将密码存储在缓存文件中。如果 `seed.dat` 文件丢失或损坏，数据库密码可能会错误地不进行加密。在这种情况下，连接将会失败。

将项目的 UNION ALL 视图与分区表结合使用

如果源表已分区，则 AWS SCT 会创建 n 个目标表，其中 n 是源表上的分区数。AWS SCT 在目标表顶部创建一个 UNION ALL 视图来表示源表。如果使用 AWS SCT 数据提取器迁移数据，源表分区将由单独的子任务并行提取和加载。

使用项目的 Union All 视图

1. 启动 AWS SCT。创建新项目，或打开现有 AWS SCT 项目。
2. 在设置 菜单上，选择转换设置。
3. 从顶部的列表中选择一对 OLAP 数据库。
4. 开启使用 Union all 视图？



5. 选择确定以保存设置并关闭转换设置对话框。

AWS SCT 的键盘快捷键

以下是您可以用于 AWS SCT 的键盘快捷键。

键盘快捷键	描述
Ctrl+N	创建新项目。

键盘快捷键	描述
Ctrl+O	打开现有项目。
Ctrl+S	保存打开的项目。
Ctrl+W	使用向导创建新项目。
Ctrl+M	创建新的多服务器评估。
Ctrl+L	添加新的源数据库。
Ctrl+R	添加新的目标数据库。
Ctrl+F4	关闭打开的项目。
F1	打开《AWS SCT 用户指南》

AWS SCT 入门

可以使用 AWS Schema Conversion Tool（AWS SCT）转换源数据库的架构。源数据库可以是自管理引擎，可以在本地运行，也可以在 Amazon EC2 实例上运行。您可以将源架构转换为由 AWS 托管的任何支持的数据库的架构。AWS SCT 应用程序提供基于项目的用户界面。

几乎所有使用 AWS SCT 执行的工作都从以下步骤开始：

1. 安装 AWS SCT。有关更多信息，请参阅安装、验证和更新 AWS SCT。
2. 根据需要安装 AWS SCT 代理。只有某些迁移方案（例如异构源和目标之间的迁移）才需要 AWS SCT 代理。有关更多信息，请参阅将本地数据仓库中的数据迁移到 Amazon Redshift。
3. 熟悉 AWS SCT 用户界面。有关更多信息，请参阅AWS SCT 用户界面的使用。
4. 创建 AWS SCT 项目。连接到您的源数据库和目标数据库。有关连接到源数据库的更多信息，请参阅AWS SCT 的源。
5. 创建映射规则。有关映射规则的更多信息，请参阅 在 AWS SCT 中创建映射规则。
6. 运行，然后查看数据库迁移评估报告。有关评估报告的更多信息，请参阅创建和查看数据库迁移评估报告。
7. 转换源数据库架构。转换有几个方面是需要记住的，例如如何对待不转换的项目，以及如何映射应按特定方式转换的项目。有关转换源架构的更多信息，请参阅使用 AWS SCT 转换数据库架构。

   如果您要转换数据仓库架构，则在进行转换之前还有一些方面需要考虑。有关更多信息，请参阅使用 AWS SCT 将数据仓库架构转换为 Amazon Redshift。
8. 将架构转换应用于您的目标。有关应用源架构转换的更多信息，请参阅应用转换后的代码。
9. 您可以使用 AWS SCT 转换 SQL 存储过程和其他应用程序代码。有关更多信息，请参阅使用 AWS SCT 转换应用程序 SQL。

您还可以使用 AWS SCT 将数据从源数据库迁移到 Amazon 管理的数据库。有关示例，请参阅 将本地数据仓库中的数据迁移到 Amazon Redshift。

AWS SCT 的源

AWS Schema Conversion Tool (AWS SCT) 可以将架构从以下源数据库和数据仓库转换为目标数据库或数据仓库。有关权限、连接以及 AWS SCT 可转换的用于目标数据库或数据仓库的内容的信息，请参阅以下主题中的详细信息。

加密信息

加密 Amazon RDS 连接

数据库源

- 使用 Apache Cassandra 作为源
- 将 Azure SQL 数据库作为源
- 将 IBM Db2 for z/OS 用作源
- 将 IBM Db2 LUW 作为源
- 将 MySQL 作为源
- 将 Oracle 数据库作为源
- 将 PostgreSQL 作为源
- 使用 SAP ASE (Sybase ASE) 作为源
- 将 SQL Server 作为源

数据仓库源

- 将 Amazon Redshift 用作源
- 使用 Azure Synapse Analytics 作为源
- 使用 BigQuery 作源
- 将 Greenplum 数据库用作源
- 将 Netezza 用作源
- 将 Oracle 数据仓库用作源
- 使用 Snowflake 作为源
- 将 SQL Server 数据仓库用作源
- 将 Teradata 用作源

- 将 Vertica 用作源

大数据来源

- 使用 Apache Hadoop 作为源
- 使用 Apache Oozie 作为源

在 AWS SCT 中加密 Amazon RDS 和 Amazon Aurora 连接

要从应用程序打开与 Amazon RDS 或 Amazon Aurora 数据库的加密连接，您需要将 AWS 根证书导入某种形式的密钥存储中。您可以从 AWS 下载根证书，具体请参阅《Amazon RDS 用户指南》中的使用 SSL/TLS 加密与数据库实例的连接。

有两个选项可用：适用于所有 AWS 区域的根证书以及同时包含新旧根证书的证书捆绑包。

根据您要使用的服务，请按照以下两个过程之一的步骤操作。

将一个或多个证书导入 Windows 系统存储器

1. 从以下来源之一下载一个或多个证书：

   有关下载证书的信息，请参阅《Amazon RDS 用户指南》中的使用 SSL/TLS 加密与数据库实例的连接。
2. 在 Windows 搜索窗口中，输入 **Manage computer certificates**。当系统提示是否允许应用程序对您的计算机进行更改时，选择是。
3. 当证书窗口打开时，如果需要，请展开证书 - 本地计算机，以便看到证书列表。打开受信任的根证书颁发机构的上下文（右键单击）菜单，然后选择所有任务和导入。
4. 选择下一步，然后选择浏览，再找到您在步骤 1 中下载的 `*.pem` 文件。选择打开以选择证书文件，然后选择下一步，再选择完成。

   > **Note**
   >
   > 要找到文件，在浏览窗口中将文件类型更改为所有文件 (*.*)，因为 `.pem` 不是标准证书扩展名。

5. 在 Microsoft 管理控制台中，展开证书。然后展开受信任的根证书颁发机构，选择证书，找到证书以确认其存在。证书的名称以 `Amazon RDS` 开头。

6. 重新启动您的计算机。

将一个或多个证书导入 Java 密钥库

1. 从以下来源之一下载一个或多个证书：

   有关下载证书的信息，请参阅《Amazon RDS 用户指南》中的使用 SSL/TLS 加密与数据库实例的连接。

2. 如果已下载了证书捆绑包，请将其拆分为单独的证书文件。为此，请将每个证书块（以 -----BEGIN CERTIFICATE----- 开头和以 -----END CERTIFICATE----- 结尾）放入单独的 *.pem 文件中。为每个证书创建单独的 *.pem 文件后，您可以安全地删除证书包捆绑包文件。

3. 在您下载证书的目录中打开命令窗口或终端会话，然后对上一步中创建的每个 *.pem 文件运行以下命令。

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

   Example

   以下示例假定您已下载 eu-west-1-bundle.pem 文件。

   ```
   keytool -importcert -file eu-west-1-bundle.pem -alias eu-west-1-bundle.pem -
   keystore trust-2019.ks
   Picked up JAVA_TOOL_OPTIONS: -Dlog4j2.formatMsgNoLookups=true
   Enter keystore password:
   Re-enter new password:
   Owner: CN=Amazon RDS Root 2019 CA, OU=Amazon RDS, O="Amazon Web Services, Inc.",
    ST=Washington, L=Seattle, C=US
   Issuer: CN=Amazon RDS Root 2019 CA, OU=Amazon RDS, O="Amazon Web Services, Inc.",
    ST=Washington, L=Seattle, C=US
   Serial number: c73467369250ae75
   Valid from: Thu Aug 22 19:08:50 CEST 2019 until: Thu Aug 22 19:08:50 CEST 2024
   Certificate fingerprints:
            SHA1: D4:0D:DB:29:E3:75:0D:FF:A6:71:C3:14:0B:BF:5F:47:8D:1C:80:96
            SHA256:
    F2:54:C7:D5:E9:23:B5:B7:51:0C:D7:9E:F7:77:7C:1C:A7:E6:4A:3C:97:22:E4:0D:64:54:78:FC:70:AA:
   Signature algorithm name: SHA256withRSA
   Subject Public Key Algorithm: 2048-bit RSA key
   Version: 3

   Extensions:
   ```

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

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

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

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

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

4. 将密钥库作为信任存储添加到 AWS SCT 中。为此，请从主菜单中选择设置、全局设置、安全、信任存储，然后选择选择现有信任存储。

   添加信任存储后，可以在创建数据库 AWS SCT 连接时使用信任存储配置启用 SSL 的连接。在 AWS SCT 连接到数据库对话框中，选择使用 SSL，然后选择之前输入的信任存储。

使用 Apache Cassandra 作为 AWS SCT 的源

您可以使用 AWS SCT 将密钥空间从 Apache Cassandra 转换到 Amazon DynamoDB。

连接到作为源的 Apache Cassandra

使用 AWS Schema Conversion Tool 按照以下过程连接到 Apache Cassandra 源数据库。

连接到 Apache Cassandra 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Cassandra，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅 使用 AWS Secrets Manager。
   - 要手动输入 Apache Cassandra 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名服务 (DNS) 名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	如果要使用安全套接字层（SSL）连接到数据库，请选择此选项。在 SSL 选项卡上提供以下其他信息（如适用）： • 信任存储：要使用的信任存储。

参数	操作
	• 密钥存储：要使用的密钥存储。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

使用 Apache Hadoop 作为 AWS SCT 的源

您可使用 AWS SCT 命令行界面（CLI）从 Apache Hadoop 迁移到 Amazon EMR。AWS SCT 在迁移期间将 Amazon S3 存储桶作为数据的临时存储空间。

AWS SCT 支持作为源的 Apache Hadoop 版本 2.2.0 及更高版本。此外，AWS SCT 还支持 Apache Hive 版本 0.13.0 及更高版本。

AWS SCT 支持作为目标的 Amazon EMR 版本 6.3.0 及更高版本。此外，AWS SCT 还支持作为目标的 Apache Hadoop 版本 2.6.0 及更高版本以及 Apache Hive 0.13.0 及更高版本。

主题

使用 Apache Hadoop 为源的先决条件

使用 AWS SCT CLI 连接到 Apache Hadoop 需要满足以下先决条件。

- 创建 Amazon S3 存储桶以在迁移期间存储数据。然后，您可以将数据复制到 Amazon EMR HDFS 或使用 Amazon S3 作为 Hadoop 工作负载的数据存储库。有关更多信息，请参阅《Amazon S3 用户指南》中的创建存储桶。
- 使用 AmazonS3FullAccess 策略创建 AWS Identity and Access Management (IAM) 角色。AWS SCT 使用此 IAM 角色访问 Amazon S3 存储桶。
- 记下您的 AWS 私有密钥和 AWS 秘密访问密钥。有关 AWS 访问密钥的更多信息，请参阅《IAM 用户指南》中的管理访问密钥。
- 创建和配置目标 Amazon EMR 集群。有关的更多信息，请参阅《Amazon EMR 管理指南》中的 Amazon EMR 入门。
- 在源 Apache Hadoop 集群上安装 distcp 实用程序。此外，在目标 Amazon EMR 集群上安装 s3-dist-cp 实用程序。确保数据库用户拥有运行这些实用程序的权限。
- 将源 Hadoop 集群中的 core-site.xml 文件配置为使用 s3a 协议。为此，将 fs.s3a.aws.credentials.provider 参数设置为以下值之一。
  - org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider
  - org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider
  - org.apache.hadoop.fs.s3a.AnonymousAWSCredentialsProvider
  - org.apache.hadoop.fs.s3a.auth.AssumedRoleCredentialProvider

  您可以将以下代码示例添加到 core-site.xml 文件中。

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

  前面的示例显示了前面选项列表中的四个选项之一。如果您未在 core-site.xml 文件中设置 fs.s3a.aws.credentials.provider 参数，则 AWS SCT 会自动选择提供程序。

使用 Hive 作为源的权限

Hive 源用户所需的权限如下：

- READ 访问源数据文件夹和源 Amazon S3 存储桶
- READ+WRITE 访问中间 Amazon S3 存储桶和目标 Amazon S3 存储桶

为了提高迁移速度，建议您运行 ACID 事务源表压缩。

Amazon EMR Hive 目标用户所需的权限如下：

- READ 访问目标 Amazon S3 存储桶。
- READ+WRITE 访问中间 Amazon S3 存储桶
- READ+WRITE 访问目标 HDFS 文件夹

使用 HDFS 作为源的权限

HDFS 作为源所需的权限如下：

- EXECUTE 适用 NameNode
- EXECUTE+READ 适用于迁移项目中包含的所有源文件夹和文件
- READ+WRITE 适用于迁移到 Amazon S3 之前运行 Spark 作业和存储文件的 NameNode 中的 tmp 目录

在 HDFS 中，所有操作都需要遍历访问权限。遍历访问需要路径中所有现有组件的 EXECUTE 权限，但最终路径组件除外。例如，对于访问 /foo/bar/baz 的任何操作，您的用户都必须拥有 /、/foo 和 /foo/bar 的 EXECUTE 权限。

以下代码示例演示如何授予源文件夹和文件 EXECUTE+READ 权限以及 tmp 目录的 READ+WRITE 权限。

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

使用 HDFS 作为目标的权限

将 Amazon EMR HDFS 作为目标所需的权限如下：

- EXECUTE 用于目标 Amazon EMR 集群的 NameNode
- READ+WRITE 用于迁移后存储数据的目标 HDFS 文件夹

连接到作为源的 Apache Hadoop

在 AWS SCT 1.0.670 或更高版本中，可以将 Apache Hadoop 作为源。您只能通过 AWS SCT 命令行界面（CLI）将 Hadoop 集群迁移到 Amazon EMR。在开始之前，请熟悉 AWS SCT 的命令行界面。有关更多信息，请参阅AWS SCT CLI 参考。

在 AWS SCT CLI 中连接到 Apache Hadoop

1. 创建新的 AWS SCT CLI 脚本或编辑现有场景模板。例如，您可以下载和编辑 `HadoopMigrationTemplate.scts` 模板。有关更多信息，请参阅获取 CLI 场景。
2. 配置 AWS SCT 应用程序设置，例如驱动程序位置和日志文件夹。

   下载所需的 JDBC 驱动程序并指定存储文件的位置。有关更多信息，请参阅下载所需的数据库驱动程序。

   以下代码示例显示如何将路径添加到 Apache Hive 驱动程序。运行此代码示例后，AWS SCT 将日志文件存储在 `c:\sct` 文件夹中。

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

   你可以在 Windows 中使用此示例和以下示例。
3. 创建新 AWS SCT 项目

   以下代码示例在 `c:\sct` 文件夹中创建 `hadoop_emr` 项目。

   ```
   CreateProject
       -name: 'hadoop_emr'
       -directory: 'c:\sct'
   /
   ```
4. 将源 Hadoop 集群添加到项目中。

使用 AddSourceCluster 命令连接到源 Hadoop 集群。请确保为以下必填参数提供值：name、host、port 和 user。其他参数都是可选的。

以下代码示例添加了源 Hadoop 集群。此示例将 HADOOP_SOURCE 设置为源集群的名称。使用此对象名称将 Hive 和 HDFS 服务添加到项目并创建映射规则。

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

在前面的示例中，将 *hadoop_address* 替换为 Hadoop 集群的 IP 地址。如果需要，请配置端口选项的值。接下来，将 *hadoop_user* 和 *hadoop_password* 替换为 Hadoop 用户的名字和该用户的密码。在*##\##*中，输入源 Hadoop 集群的 PEM 文件的名称和路径。

5. 保存 CLI 脚本。接下来，添加 Hive 和 HDFS 服务的连接信息。

连接到源 Hive 和 HDFS 服务

您可以使用 AWS SCT CLI 连接到源 Hive 和 HDFS 服务。要连接到 Apache Hive，请使用 Hive JDBC 驱动程序版本 2.3.4 或更高版本。有关更多信息，请参阅下载所需的数据库驱动程序。

AWS SCT 通过 hadoop 集群用户连接到 Apache Hive。为此，请使用 AddSourceClusterHive 和 AddSourceClusterHDFS 命令。您可以使用以下方法之一。

- 创建一个新的 SSH 隧道。

  对于 createTunnel，输入 **true**。对于 host，请输入源 Hive 或 HDFS 服务的内部 IP 地址。对于 port，请输入 Hive 或 HDFS 服务的服务端口。

  接下来，输入 user 和 password 的 Hive 或 HDFS 凭证。有关 SSH 隧道的更多信息，请参阅《Amazon EMR 管理指南》中的使用本地端口转发设置到主节点的 SSH 隧道。
- 使用现有的 SSH 隧道。

对于 host，输入 **localhost**。对于 port，从 SSH 隧道参数中输入本地端口。

- 直接连接到 Hive 和 HDFS 服务。

  对于 host，请输入源 Hive 或 HDFS 服务的 IP 地址或主机名。对于 port，请输入 Hive 或 HDFS 服务的服务端口。接下来，输入 user 和 password 的 Hive 或 HDFS 凭证。

在 AWS SCT CLI 中连接到 Hive 和 HDFS

1. 打开包含源 Hadoop 集群连接信息的 CLI 脚本。确保使用您在上一步中定义的 Hadoop 集群的名称。
2. 将源 Hive 服务添加到项目中。

   使用 AddSourceClusterHive 命令连接源 Hive 服务。请确保为以下必填参数提供值：user、password、cluster、name 和 port。其他参数都是可选的。

   以下代码示例创建了一个用于 AWS SCT 与 Hive 服务配合使用的隧道。此源 Hive 服务与 AWS SCT 在同一台电脑上运行。此示例使用前一个示例中的 HADOOP_SOURCE 源集群。

```
AddSourceClusterHive
    -cluster: 'HADOOP_SOURCE'
    -name: 'HIVE_SOURCE'
    -host: 'localhost'
    -port: '10005'
    -user: 'hive_user'
    -password: 'hive_password'
    -createTunnel: 'true'
    -localPort: '10005'
    -remoteHost: 'hive_remote_address'
    -remotePort: 'hive_port'
/
```

   以下代码示例无需隧道即可连接到 Hive 服务。

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

```
    -password: 'hive_password'
/
```

在前面的示例中，将 *hive_user* 和 *hive_password* 替换为 Hive 用户名和该用户的密码。

接下来，将 *hive_address* 和 *hive_port* 替换为源 Hadoop 集群的 NameNode IP 地址和端口。

对于 *hive_remote_address*，您可以使用源 Hive 服务的默认值 `127.0.0.1` 或 NameNode IP 地址。

3. 将源 HDFS 服务添加到项目中。

使用 AddSourceClusterHDFS 命令连接源 HDFS 服务。请确保为以下必填参数提供值：user、password、cluster、name 和 port。其他参数都是可选的。

确保您的用户具有从源 HDFS 服务迁移数据所需的权限。有关更多信息，请参阅使用 Hive 作为源的权限。

以下代码示例创建了一个用于 AWS SCT 与 Apache HDFS 服务配合使用的隧道。此示例使用您之前创建的 HADOOP_SOURCE 源集群。

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

以下代码无需隧道即可连接到 Apache HDFS 服务。

```
AddSourceClusterHDFS
    -cluster: 'HADOOP_SOURCE'
    -name: 'HDFS_SOURCE'
    -host: 'hdfs_address'
    -port: 'hdfs_port'
```

```
    -user: 'hdfs_user'
    -password: 'hdfs_password'
/
```

在前面的示例中，将 *hdfs_user* 和 *hdfs_password* 替换为 HDFS 用户的名称和该用户的密码。

接下来，将 *hdfs_address* 和 *hdfs_port* 替换为源 Hadoop 集群的 NameNode IP 地址和端口。

对于 *hdfs_remote_address*，您可以使用源 Hive 服务的默认值 `127.0.0.1` 或 NameNode IP 地址。

4. 保存 CLI 脚本。接下来，添加目标 Amazon EMR 集群的连接信息以及迁移命令。

连接到作为目标的 Amazon EMR

您可以使用 AWS SCT CLI 连接到目标 Amazon EMR 集群。为此，您需要授权入站流量并使用 SSH。在本例中，AWS SCT 拥有使用 Amazon EMR 集群所需的所有权限。有关更多信息，请参阅《Amazon EMR 管理指南》中的 连接之前和使用 SSH 连接到主节点。

AWS SCT 通过 hadoop 集群用户连接到 Amazon EMR Hive。要连接 Amazon EMR Hive，请使用 Hive JDBC 驱动程序版本 2.6.2.1002 或更高版本。有关更多信息，请参阅下载所需的数据库驱动程序。

在 AWS SCT CLI 中连接到 Amazon EMR

1. 打开包含源 Hadoop 集群连接信息的 CLI 脚本。将目标 Amazon EMR 凭证添加到此文件中。
2. 将目标 Amazon EMR 集群添加到项目中。

   以下代码示例添加了目标 Amazon EMR 集群。此示例将 `HADOOP_TARGET` 设置为目标集群的名称。使用此对象名称将 Hive 和 HDFS 服务以及 Amazon S3 存储桶文件夹添加到项目中，并创建映射规则。

```
AddTargetCluster
 -name: 'HADOOP_TARGET'
 -vendor: 'AMAZON_EMR'
 -host: 'ec2-44-44-55-66.eu-west-1.EXAMPLE.amazonaws.com'
 -port: '22'
 -user: 'emr_user'
```

```
 -password: 'emr_password'
 -useSSL: 'true'
 -privateKeyPath: 'c:\path\name.pem'
 -passPhrase: '1234567890abcdef0!'
 -s3Name: 'S3_TARGET'
 -accessKey: 'AKIAIOSFODNN7EXAMPLE'
 -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
 -region: 'eu-west-1'
 -s3Path: 'doc-example-bucket/example-folder'
/
```

在前面的示例中，输入 AWS 资源名称和 Amazon EMR 连接信息。这包括您的 Amazon EMR 集群的 IP 地址、AWS 访问密钥、AWS 秘密访问密钥和 Amazon S3 存储桶。如果需要，请配置端口变量的值。接下来，将 *emr_user* 和 *emr_password* 替换为 Amazon EMR 用户名和该用户的密码。在*##\##*中，输入目标 Amazon EMR 集群的 PEM 文件的名称和路径。有关更多信息，请参阅下载用于 EMR 集群访问的 PEM 文件。

3. 将目标 Amazon S3 存储桶添加到项目中。

   以下代码示例添加目标 Amazon S3 存储桶。此示例使用您之前创建的 HADOOP_TARGET 集群。

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

   在前面的示例中，输入您的 AWS 访问密钥、AWS 秘密访问密钥和 Amazon S3 存储桶。

4. 将目标 Hive 服务添加到项目中。

   以下代码示例为 AWS SCT 创建了一个隧道，以便与目标 Hive 服务配合使用。此示例使用您之前创建的 HADOOP_TARGET 目标集群。

```
AddTargetClusterHive
    -cluster: 'HADOOP_TARGET'
    -name: 'HIVE_TARGET'
    -host: 'localhost'
    -port: '10006'
```

```
    -user: 'hive_user'
    -password: 'hive_password'
    -createTunnel: 'true'
    -localPort: '10006'
    -remoteHost: 'hive_address'
    -remotePort: 'hive_port'
/
```

在前面的示例中，将 *hive_user* 和 *hive_password* 替换为 Hive 用户名和该用户的密码。

接下来，将 *hive_address* 替换为默认值 127.0.0.1 或目标 Hive 服务的 NameNode IP 地址。接下来，将 *hive_port* 替换为目标 Hive 服务的端口。

5. 将目标 HDFS 服务添加到项目中。

以下代码示例创建了一个用于 AWS SCT 与 Apache HDFS 服务配合使用的隧道。此示例使用您之前创建的 HADOOP_TARGET 目标集群。

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

在前面的示例中，将 *hdfs_user* 和 *hdfs_password* 替换为 HDFS 用户的名称和该用户的密码。

接下来，将 *hdfs_address* 和 *hdfs_port* 替换为目标 HDFS 服务的 NameNode 的私有 IP 地址和端口。

6. 保存 CLI 脚本。接下来，添加映射规则和迁移命令。有关更多信息，请参阅将 Apache Hadoop 迁移到 Amazon EMR。

使用 Apache Oozie 作为 AWS SCT 的源

您可使用 AWS SCT 命令行界面（CLI）将 Apache Oozie 工作流程转换为 AWS Step Functions。将 Apache Hadoop 工作负载迁移到 Amazon EMR 后，您可以使用 AWS Cloud 中的本地服务编排任务。有关更多信息，请参阅使用 Apache Hadoop 作为源。

AWS SCT 将 Oozie 工作流程转换为 AWS Step Functions，并使用 AWS Lambda 模拟 AWS Step Functions 不支持的功能。此外，AWS SCT 还可以将 Oozie 作业属性转换为 AWS Systems Manager。

要转换 Apache Oozie 工作流程，请确保使用 AWS SCT 版本 1.0.671 或更高版本。另外，请熟悉 AWS SCT 的命令行界面。有关更多信息，请参阅AWS SCT CLI 参考。

使用 Apache Oozie 为源的先决条件

使用 CLI AWS SCT 连接到 Apache Oozie 需要满足以下先决条件。

- 创建 Amazon S3 存储桶以存储状态机的定义。您可以使用这些定义配置状态机。有关更多信息，请参阅《Amazon S3 用户指南》中的创建存储桶。
- 使用 `AmazonS3FullAccess` 策略创建 AWS Identity and Access Management (IAM) 角色。AWS SCT 使用此 IAM 角色访问 Amazon S3 存储桶。
- 记下您的 AWS 私有密钥和 AWS 秘密访问密钥。有关 AWS 访问密钥的更多信息，请参阅《IAM 用户指南》中的管理访问密钥。
- 在全局应用程序设置中将 AWS 凭证和有关 Amazon S3 存储桶的信息存储到 AWS 服务配置文件中。然后，AWS SCT 使用此 AWS 服务配置文件处理 AWS 资源。有关更多信息，请参阅将 AWS 服务配置文件存储在 AWS SCT 中。

要使用源代码 Apache Oozie 工作流程，AWS SCT 需要源文件的特定结构。每个应用程序文件夹都必须包含 `job.properties` 文件。此文件包含任务属性的键值对。此外，每个应用程序文件夹都必须包含 `workflow.xml` 文件。此文件描述了工作流程的操作节点和控制流节点。

连接到作为源的 Apache Oozie

按照以下过程连接到 Apache Oozie 源文件。

在 AWS SCT CLI 中连接到 Apache Oozie

1. 创建新的 AWS SCT CLI 脚本或编辑现有场景模板。例如，您可以下载和编辑 `OozieConversionTemplate.scts` 模板。有关更多信息，请参阅获取 CLI 场景。

2. 配置 AWS SCT 应用程序设置。

   以下代码示例保存了应用程序设置并允许在项目中存储密码。您可以在其他项目中使用这些保存的设置。

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

3. 创建新 AWS SCT 项目

   以下代码示例在 c:\sct 文件夹中创建 oozie 项目。

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

4. 使用 AddSource 命令将包含源 Apache Oozie 文件的文件夹添加到项目中。确保使用 vendor 参数的 APACHE_OOZIE 值。此外，还要提供以下必需参数的值： name 和 mappingsFolder。

   以下代码示例将作为源的 Apache Oozie 添加到 AWS SCT 项目中。此示例创建了一个名为 OOZIE 的源对象。使用此对象名称添加映射规则。运行此代码示例后，AWS SCT 使用 c:\oozie 文件夹将源文件加载到项目中。

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

   你可以在 Windows 中使用此示例和以下示例。

5. 使用 ConnectSource 命令连接到源 Apache Oozie 文件。使用在上一步骤中定义的源对象的名称。

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

```
/
```

6. 保存 CLI 脚本。接下来，请为 AWS Step Functions 服务添加连接信息。

使用扩展包中 AWS Lambda 函数的权限

对于 AWS Step Functions 不支持的源函数，AWS SCT 创建一个扩展包。此扩展包包含模拟源函数的 AWS Lambda 函数。

要使用此扩展包，请创建具有以下权限的 AWS Identity and Access Management（IAM）角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "lambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:498160209112:function:LoadParameterInitialState:*",
                "arn:aws:lambda:*:498160209112:function:EvaluateJSPELExpressions:*"
            ]
        },
        {
            "Sid": "emr",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:AddJobFlowSteps"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:*:498160209112:cluster/*"
            ]
        },
        {
            "Sid": "s3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::*/*"
            ]
        }
    ]
}
```

要应用扩展包，AWS SCT需要具有以下权限的 IAM 角色。

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

```
            "Action": [
                "lambda:GetFunction",
                "lambda:CreateFunction",
                "lambda:UpdateFunctionCode",
                "lambda:DeleteFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:ACCOUNT_NUMBER:function:LoadParameterInitialState",
                "arn:aws:lambda:*:ACCOUNT_NUMBER:function:EvaluateJSPELExpressions"
            ]
        }
    ]
}
```

连接到作为目标的 AWS Step Functions

使用以下过程连接到作为目标的 AWS Step Functions。

在 AWS SCT CLI 中连接到 AWS Step Functions

1. 打开包含 Apache Oozie 源文件的连接信息的 CLI 脚本。
2. 使用 `AddTarget` 命令在 AWS SCT 项目中添加有关迁移目标的信息。确保使用 `vendor` 参数的 `STEP_FUNCTIONS` 值。此外，还要提供以下必需参数的值： `name` 和 `profile`。

   以下代码示例将作为源的 AWS Step Functions 添加到 AWS SCT 项目中。此示例创建了一个名为 `AWS_STEP_FUNCTIONS` 的目标对象。创建映射规则时使用此对象名称。此外，此示例还使用您在先决条件步骤中创建的 AWS SCT 服务配置文件。请务必使用配置名称替换 *`profile_name`*。

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

   如果不使用 AWS 服务配置文件，请确保为以下必需参数提供值：`accessKey`、`secretKey`、`awsRegion` 和 `s3Path`。使用这些参数指定 AWS 秘密访问密钥、AWS 私有密钥、AWS 区域 和 Amazon S3 存储桶的路径。
3. 使用 `ConnectTarget` 命令，连接到 AWS Step Functions。使用在上一步骤中定义的目标对象的名称。

以下代码示例使用 AWS 服务配置文件连接到 AWS_STEP_FUNCTIONS 目标对象。请务必使用配置名称替换 *profile_name*。

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

4. 保存 CLI 脚本。接下来，添加映射规则和迁移命令。有关更多信息，请参阅将 Apache Oozie 转换为 AWS Step Functions。

将 Azure SQL 数据库作为 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Azure SQL Database 转换到以下目标：

- Amazon RDS for MySQL
- Amazon Aurora MySQL 兼容版
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版

主题

- 将 Azure SQL 数据库用作源的权限
- 连接到作为源的 Azure SQL 数据库

将 Azure SQL 数据库用作源的权限

将 Azure SQL 数据库作为源所需的权限如下：

- VIEW DEFINITION
- VIEW DATABASE STATE

对要转换其架构的每个数据库重复这种授权。

以下部分将介绍目标 MySQL 和 PostgreSQL 数据库所需的权限。

- 将 MySQL 用作目标数据库的权限

- 将 PostgreSQL 用作目标数据库的权限

连接到作为源的 Azure SQL 数据库

使用 AWS Schema Conversion Tool 按照以下过程连接到 Azure SQL 数据库源数据库。

连接到 Azure SQL 数据库源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Azure SQL 数据库，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 Azure SQL 数据库源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名服务 (DNS) 名称或 IP 地址。
数据库。	输入要连接的数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。

参数	操作
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 IBM Db2 for z/OS 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 IBM Db2 for z/OS 转换到以下目标：

- Amazon RDS for MySQL
- Amazon Aurora MySQL 兼容版
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版

将 Db2 for z/OS 用作源数据库的先决条件

IBM Db2 for z/OS 版本 12 函数级别 100 数据库版本不支持 IBM Db2 for z/OS 版本 12 的大多数新功能。此数据库版本支持回退到 Db2 版本 11 以及与 Db2 版本 11 的数据共享。为避免转换 Db2 版本 11 中不支持的功能，建议您使用 IBM Db2 for z/OS 数据库函数级别 500 或更高版本作为 AWS SCT 源。

您可以使用以下代码示例检查源 IBM Db2 for z/OS 数据库的版本。

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

请确保此代码返回版本 DSN12015 或更高版本。

您可以使用以下代码示例检查源 IBM Db2 for z/OS 数据库中 APPLICATION COMPATIBILITY 特殊寄存器的值。

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

请确保此代码返回版本 V12R1M500 或更高版本。

将 Db2 for z/OS 用作源数据库的权限

连接到 Db2 for z/OS 数据库以及读取系统目录和表所需的权限如下：

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

要将 Db2 for z/OS 表转换为 PostgreSQL 分区表，请使用 RUNSTATS 实用程序收集数据库中表空间和表的统计信息，如下所示。

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

在上述示例中，将 *YOURDB* 占位符替换为源数据库的名称。

连接到作为源的 Db2 for z/OS

使用 AWS SCT 按照以下过程连接到 Db2 for z/OS 源数据库。

连接到 IBM Db2 for z/OS 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Db2 for z/OS，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：
     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 IBM Db2 for z/OS 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
位置	输入要访问的 Db2 位置的唯一名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。

参数	操作
	仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	如果要使用安全套接字层 (SSL) 连接到数据库，请选择此选项。在 SSL 选项卡上提供以下其他信息（如适用）： • 信任存储：包含证书的信任存储的位置。要使此位置出现在此处，请务必将其添加到全局设置中。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
Db2 for z/OS 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 MySQL 用作目标数据库的权限

下面列出了将 MySQL 用作目标所需的权限：

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

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

您可以使用以下代码示例创建数据库用户并授予权限。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT SELECT ON mysql.proc TO 'user_name';
GRANT INSERT, UPDATE ON AWS_DB2ZOS_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_DB2ZOS_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_DB2ZOS_EXT_DATA.* TO 'user_name';
```

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *your_password* 替换为安全密码。

要使用 Amazon RDS for MySQL 作为目标，请将 `log_bin_trust_function_creators` 参数设置为 true，将 `character_set_server` 设置为 `latin1`。要配置这些参数，请创建新的数据库参数组或修改现有数据库参数组。

要使用 Aurora MySQL 作为目标，请将 log_bin_trust_function_creators 参数设置为 true，将 character_set_server 设置为 latin1。此外，还要将 lower_case_table_names 参数设置为 True。要配置这些参数，请创建新的数据库参数组或修改现有数据库参数组。

将 PostgreSQL 用作目标数据库的权限

要使用 PostgreSQL 作为目标，AWS SCT 需要 CREATE ON DATABASE 权限。请确保为每个目标 PostgreSQL 数据库授予此权限。

要使用 Amazon RDS for PostgreSQL 作为目标，AWS SCT 需要 rds_superuser 权限。

要使用转换后的公共同义词，请将数据库的默认搜索路径更改为 "$user", public_synonyms, public。

您可以使用以下代码示例创建数据库用户并授予权限。

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *db_name* 替换为目标数据库名称。最后，将 *your_password* 替换为安全密码。

在 PostgreSQL 中，只有架构所有者或 superuser 才能删除架构。即使架构的所有者并不拥有架构的某些对象，该所有者也可以删除该架构及其包含的所有对象。

使用不同的用户转换不同的架构并将其应用到目标数据库时，若 AWS SCT 无法删除架构，您可能会收到一条错误消息。要避免出现此错误消息，请使用 superuser 角色。

Db2 for z/OS 到 PostgreSQL 的转换设置

要编辑 Db2 for z/OS 到 PostgreSQL 的转换设置，请选择设置，然后选择转换设置。从上面的列表中选择 Db2 for z/OS，然后选择 Db2 for z/OS – PostgreSQL 或 Db2 for z/OS – Amazon Aurora（兼容 PostgreSQL）。AWS SCT 显示 IBM Db2 for z/OS 到 PostgreSQL 转换的所有可用设置。

AWS SCT 中的 Db2 for z/OS 到 PostgreSQL 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 为目标数据库中的约束生成唯一名称。

  在 PostgreSQL 中，您使用的所有约束名称都必须是唯一的。通过在约束名称中添加带有表名的前缀，AWS SCT 可以在转换后的代码中为约束生成唯一的名称。要确保 AWS SCT 为约束生成唯一的名称，请选择为约束生成唯一名称。

- 在转换后的代码中保留 DML 语句中列名、表达式和子句的格式。

  AWS SCT 可以保持 DML 语句中列名、表达式和子句的布局与源代码中的位置和顺序相似。为此，请为保留 DML 语句中列名、表达式和子句的格式设置选择是。

- 将表分区排除在转换范围之外。

  AWS SCT 可以在转换过程中跳过源表的所有分区。为此，请选择将表分区排除在转换范围之外。

- 按增长分区的表使用自动分区。

  对于数据迁移，AWS SCT 可以自动对所有大于指定大小的表进行分区。要使用此选项，请选择强制对较大的表进行分区，然后输入以千兆字节为单位的表大小。接下来，输入分区的数量。启用此选项时，AWS SCT 会考虑源数据库的直接访问存储设备（DASD）的大小。

  AWS SCT 可以自动确定分区的数量。为此，请选择按比例增加分区数，然后输入最大分区数。

- 以 refcursor 数据类型值数组的形式返回动态结果集。

  AWS SCT 可以将返回动态结果集的源过程转换为将打开的引用游标数组作为附加输出参数的过程。为此，请选择使用引用游标数组返回所有动态结果集。

- 指定用于将日期和时间值转换为字符串表示形式的标准。

  AWS SCT 可以使用支持的行业格式之一将日期和时间值转换为字符串表示形式。为此，请选择使用日期值的字符串表示形式或使用时间值的字符串表示形式。接下来，请选择下列标准之一。

  - 国际标准组织（ISO）标准
  - IBM 欧洲标准（欧洲）
  - IBM 美国标准（美国）
  - 日本工业标准基督教时代（JIS）

将 IBM Db2 LUW 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、SQL 语言中的代码对象以及适用于 Linux、Unix 和 Windows 的 IBM Db2（Db2 LUW）中的应用程序代码转换为以下目标。

- Amazon RDS for MySQL
- Amazon Aurora MySQL 兼容版
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版
- Amazon RDS for MariaDB

AWS SCT 支持源 Db2 LUW 9.1、9.5、9.7、10.1、10.5、11.1 和 11.5 版。

将 Db2 LUW 用作源的权限

下面列出了连接到 Db2 LUW 数据库、检查可用权限和读取源的架构元数据所需的权限：

- 建立连接所需的权限：
  - CONNECT ON DATABASE
- 运行 SQL 语句所需的权限：
  - EXECUTE ON PACKAGE NULLID.SYSSH200
- 获取实例级别信息所需的权限：
  - EXECUTE ON FUNCTION SYSPROC.ENV_GET_INST_INFO
  - SELECT ON SYSIBMADM.ENV_INST_INFO
  - SELECT ON SYSIBMADM.ENV_SYS_INFO
- 检查通过角色、组和机构授予的权限所需的权限：
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID
  - SELECT ON SYSIBMADM.PRIVILEGES
- 系统目录和表上所需的权限：
  - SELECT ON SYSCAT.ATTRIBUTES
  - SELECT ON SYSCAT.CHECKS

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

- 要运行 SQL 语句，用户账户需要使用至少一个在数据库中启用的工作负载的权限。如果没有为用户分配任何工作负载，请确保用户可以访问默认用户工作负载：

- USAGE ON WORKLOAD SYSDEFAULTUSERWORKLOAD

要运行查询，您需要创建具有页面大小 8K、16K 和 32K 的系统临时表空间（如果它们不存在）。要创建临时表空间，请运行以下脚本。

```
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;
```

连接到作为源的 Db2 LUW

使用 AWS Schema Conversion Tool 按照以下过程连接到 Db2 LUW 源数据库。

连接到 Db2 LUW 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Db2 LUW，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 IBM Db2 LUW 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
数据库。	输入 Db2 LUW 数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	如果要使用安全套接字层 (SSL) 连接到数据库，请选择此选项。在 SSL 选项卡上提供以下其他信息（如适用）： • 信任存储：包含证书的信任存储的位置。要使此位置出现在此处，请务必将其添加到全局设置中。

参数	操作
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
Db2 LUW 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 Db2 LUW 转换为 Amazon RDS for PostgreSQL 或 Amazon Aurora PostgreSQL 兼容版

在将 IBM Db2 LUW 迁移到 PostgreSQL 时，AWS SCT 可以转换与 Db2 LUW 一起使用的各种触发语句。这些触发语句包括以下内容：

- 触发事件 - INSERT、DELETE 和 UPDATE 触发事件指定触发的操作将在事件应用于主题表或主题视图时运行。您可以指定 INSERT、DELETE 和 UPDATE 事件的任意组合，但每个事件只能指定一次。AWS SCT 支持单个和多个触发事件。对于事件，PostgreSQL 几乎具有相同的功能。
- 事件 OF COLUMN：可通过基表指定列名称。仅当更新列名称列表中标识的列时才会激活此触发器。PostgreSQL 具有相同的功能。
- 语句触发器：这些触发器指定仅对整个语句应用触发的操作一次。不能为 BEFORE 触发器或 INSTEAD OF 触发器指定此类型的触发器粒度。如果指定，则将激活 UPDATE 或 DELETE 触发器，即使不影响任何行也是如此。PostgreSQL 也具有此功能，并且 PostgreSQL 和 Db2 LUW 的语句触发器的触发声明是相同的。
- 引用子句：这些子句指定转换变量的相关性名称和转换表的表名称。相关性名称标识通过触发 SQL 操作影响的行集中的特定行。表名称标识受影响行的完整集。通过触发 SQL 操作影响的每行可用于通过使用指定相关性名称限定列触发的操作。PostgreSQL 不支持此功能，并且仅使用 NEW 或 OLD 相关性名称。
- INSTEAD OF 触发器：AWS SCT 支持这些触发器。

将 Db2 LUW 分区表转换为 PostgreSQL 版本 10 分区表

AWS SCT 可将 Db2 LUW 表转换为 PostgreSQL 10 中的分区表。在将 Db2 LUW 分区表转换为 PostgreSQL 时，存在一些限制：

- 可在 Db2 LUW 中创建具有可为空的列的分区表，并指定用于存储 NULL 值的分区。但是，PostgreSQL 不支持 RANGE 分区采用 NULL 值。
- Db2 LUW 可以使用 INCLUSIVE 或 EXCLUSIVE 子句来设置范围边界值。PostgreSQL 仅支持 INCLUSIVE 用于开始边界，EXCLUSIVE 用于结束边界。转换的分区名称的格式为 <original_table_name>_<original_partition_name>。
- 在 Db2 LUW 中，可为分区表创建主键或唯一键。PostgreSQL 需要您直接为每个分区创建主键或唯一键。必须从父表中删除主键或唯一键约束。转换的键名称的格式为 <original_key_name>_<original_partition _name>。
- 在 Db2 LUW 中，您可以创建进出分区表的外键约束。但是，PostgreSQL 不支持分区表中的外键引用。此外，PostgreSQL 也不支持从一个分区表到另一个表的外键引用。
- 在 Db2 LUW 中，您可以在分区表上创建索引。但是，PostgreSQL 需要您直接为每个分区创建索引。必须从父表删除索引。转换的索引名称的格式为 <original_index_name>_<original_partition_name>。
- 您必须针对单个分区而不是分区表定义行触发器。必须从父表删除触发器。转换的触发器名称的格式为 <original_trigger_name>_<original_partition_name>。

将 PostgreSQL 用作目标的权限

要使用 PostgreSQL 作为目标，AWS SCT 需要 `CREATE ON DATABASE` 权限。请确保为每个目标 PostgreSQL 数据库授予此权限。

要使用转换后的公共同义词，请将数据库的默认搜索路径更改为 `"$user", public_synonyms, public`。

您可以使用以下代码示例创建数据库用户并授予权限。

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *db_name* 替换为目标数据库名称。最后，将 *your_password* 替换为安全密码。

在 PostgreSQL 中，只有架构所有者或 superuser 才能删除架构。即使架构的所有者并不拥有架构的某些对象，该所有者也可以删除该架构及其包含的所有对象。

使用不同的用户转换不同的架构并将其应用到目标数据库时，若 AWS SCT 无法删除架构，您可能会收到一条错误消息。要避免出现此错误消息，请使用 superuser 角色。

将 Db2 LUW 转换为 Amazon RDS for MySQL 或 Amazon Aurora MySQL。

将 IBM Db2 LUW 数据库转换为适用于 MySQL 的 RDS 或 Amazon Aurora MySQL 时，请注意以下几点。

将 MySQL 用作目标的权限

下面列出了将 MySQL 用作目标所需的权限：

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

您可以使用以下代码示例创建数据库用户并授予权限。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
```

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *your_password* 替换为安全密码。

要使用 Amazon RDS for MySQL 或 Aurora MySQL 作为目标，请将 lower_case_table_names 参数设置为 1。此值意味着 MySQL 服务器在处理表、索引、触发器和数据库等对象名称的标识符时不区分大小写。如果目标实例中已开启二进制日志记录，请将 log_bin_trust_function_creators 参数设置为 1。在这种情况下，您无需使用 DETERMINISTIC、READS SQL DATA 或 NO SQL 特性创建存储函数。要配置这些参数，请创建新的数据库参数组或修改现有数据库参数组。

将 MySQL 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、数据库代码对象和应用程序代码从 MySQL 转换到以下目标：

- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版
- Amazon RDS for MySQL

有关详细信息，请参阅以下章节：

主题

- 将 MySQL 用作源数据库的权限
- 连接到作为源的 MySQL
- 将 PostgreSQL 用作目标数据库的权限

将 MySQL 用作源数据库的权限

下面是将 MySQL 用作源所需的权限：

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

连接到作为源的 MySQL

使用 AWS Schema Conversion Tool 按照以下过程连接到 MySQL 源数据库。

连接到 MySQL 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 MySQL，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：
     1. 对于 AWS 密钥，请选择密钥的名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅 使用 AWS Secrets Manager。
   - 要手动输入 MySQL 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。 您可以使用 IPv6 地址协议连接到源 MySQL 数据库。为此，请确保使用方括号输入 IP 地址，如以下示例所示。 `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
服务器端口	输入用于连接到源数据库服务器的端口。

参数	操作
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层（SSL）连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 需要 SSL：选择此选项仅通过 SSL 连接到服务器。 如果您选择 Require SSL，则意味着如果服务器不支持 SSL，您将无法连接到服务器。如果您不选择 Require SSL 且服务器不支持 SSL，您仍然可以在不使用 SSL 的情况下连接到服务器。有关更多信息，请参阅配置 MySQL 以使用安全连接。 • 验证服务器证书：选择此选项以使用信任存储验证服务器证书。 • 信任存储：包含证书的信任存储的位置。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项可让您存储数据库密码并在不需要输入密码的情况下快速连接到数据库。
MySql 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 PostgreSQL 用作目标数据库的权限

要使用 PostgreSQL 作为目标，AWS SCT 需要 CREATE ON DATABASE 权限。请确保为每个目标 PostgreSQL 数据库授予此权限。

要使用转换后的公共同义词，请将数据库的默认搜索路径更改为 "$user", public_synonyms, public。

您可以使用以下代码示例创建数据库用户并授予权限。

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *db_name* 替换为目标数据库名称。最后，将 *your_password* 替换为安全密码。

在 PostgreSQL 中，只有架构所有者或 superuser 才能删除架构。即使架构的所有者并不拥有架构的某些对象，该所有者也可以删除该架构及其包含的所有对象。

使用不同的用户转换不同的架构并将其应用到目标数据库时，若 AWS SCT 无法删除架构，您可能会收到一条错误消息。要避免出现此错误消息，请使用 superuser 角色。

将 Oracle 数据库作为 AWS SCT 的源

您可以使用 AWS SCT 将架构、数据库代码对象和应用程序代码从 Oracle 数据库转换到以下目标：

- Amazon RDS for MySQL
- Amazon Aurora MySQL 兼容版
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版
- Amazon RDS for Oracle
- Amazon RDS for MariaDB

当源是 Oracle 数据库时，可以将注释转换为 PostgreSQL 等数据库中适当的格式。AWS SCT 可以转换表、视图和列上的注释。注释可以包括撇号；转换 SQL 语句时，AWS SCT 会将撇号加倍，就像字符串文本一样。

有关更多信息，请参阅下列内容。

主题

将 Oracle 用作源的权限

下面列出了将 Oracle 用作源所需的权限：

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

连接到作为源的 Oracle

使用 AWS Schema Conversion Tool 按照以下过程连接到 Oracle 源数据库。

连接到 Oracle 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源 。
2. 选择 Oracle，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，请选择密钥的名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅 使用 AWS Secrets Manager。
   - 要手动输入 Oracle 源数据库连接信息，请按照以下说明进行操作：

参数	操作

Type	选择连接到您的数据库的连接类型。根据类型，提供以下附加信息：
	• SID
	  • 服务器名称：源数据库服务器的域名系统（DNS）名称或 IP 地址。
	  • 服务器端口：键入用于连接到源数据库服务器的端口。
	  • Oracle SID：Oracle 系统 ID (SID)。要查找 Oracle SID，请向您的 Oracle 数据库提交以下查询：
	`SELECT sys_context('userenv','instance_name') AS SID FROM dual;`
	• 服务名称
	  • Server name：键入源数据库服务器的 DNS 名称或 IP 地址。
	您可以使用 IPv6 地址协议连接到源 Oracle 数据库。为此，请确保使用方括号输入 IP 地址，如以下示例所示。
	`[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
	  • 服务器端口：键入用于连接到源数据库服务器的端口。
	  • 服务名称：要连接到的 Oracle 服务的名称。
	• TNS 别名
	  • TNS file path：包含透明网络底层 (TNS) 名称连接信息的文件的路径。
	选择 TNS 文件后，AWS SCT 会将文件中的所有 Oracle 数据库连接添加到 TNS 别名列表中。
	选择此选项可连接到 Oracle Real Application Clusters（RAC）。
	  • TNS 别名：用于连接到源数据库的此文件中的 TNS 别名。

参数	操作
	• TNS 连接标识符 • TNS 标识符：已注册 TNS 连接信息的标识符。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 首次连接到 Oracle 数据库时，您需要输入到 Oracle 驱动程序文件（ojdbc8.jar）的路径。您可以在 http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html 中下载该文件。请务必在免费的 Oracle 技术网络网站上注册以完成下载。AWS SCT 将选定的驱动程序用于将来的任何 Oracle 数据库连接。可以使用全局设置中的驱动程序选项卡修改驱动程序路径。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层（SSL）连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • SSL 身份验证：选择此选项可通过证书使用 SSL 身份验证。在设置、全局设置、安全中设置您的信任存储和密钥存储。 • 信任存储：要使用的信任存储。 • 密钥存储：要使用的密钥存储。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。选择此选项可存储数据库密码并在不需要输入密码的情况下快速连接到数据库。

参数	操作
Oracle 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 Oracle 转换为 Amazon RDS for PostgreSQL 或 Amazon Aurora PostgreSQL

将 Oracle 数据库转换为适用于 PostgreSQL 的 RDS 或 Amazon Aurora PostgreSQL 时，请注意以下几点。

主题

- 将 PostgreSQL 用作目标数据库的权限
- Oracle 到 PostgreSQL 的转换设置
- 转换 Oracle 序列
- 转换 Oracle ROWID
- 转换 Oracle 动态 SQL
- 转换 Oracle 分区

将 Oracle 系统对象转换为 PostgreSQL 时，AWS SCT 按下表所示执行转换。

Oracle 系统对象	描述	转换的 PostgreSQL 对象
V$VERSION	在 Oracle 数据库中显示核心库组件的版本号	aws_oracle_ext.v$version

Oracle 系统对象	描述	转换的 PostgreSQL 对象
V$INSTANCE	显示当前实例状态的视图。	aws_oracle_ext.v$instance

您可以使用 AWS SCT 将 Oracle SQL*Plus 文件转换为 psql，psql 是 PostgreSQL 的基于终端的前端。有关更多信息，请参阅使用 AWS SCT 转换应用程序 SQL。

将 PostgreSQL 用作目标数据库的权限

要使用 PostgreSQL 作为目标，AWS SCT 需要 `CREATE ON DATABASE` 权限。请确保为每个目标 PostgreSQL 数据库授予此权限。

要使用转换后的公共同义词，请将数据库的默认搜索路径更改为 `"$user", public_synonyms, public`。

您可以使用以下代码示例创建数据库用户并授予权限。

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *db_name* 替换为目标数据库名称。最后，将 *your_password* 替换为安全密码。

要使用 Amazon RDS for PostgreSQL 作为目标，AWS SCT 需要 `rds_superuser` 权限。

在 PostgreSQL 中，只有架构所有者或 `superuser` 才能删除架构。即使架构的所有者并不拥有架构的某些对象，该所有者也可以删除该架构及其包含的所有对象。

使用不同的用户转换不同的架构并将其应用到目标数据库时，若 AWS SCT 无法删除架构，您可能会收到一条错误消息。要避免出现此错误消息，请使用 `superuser` 角色。

Oracle 到 PostgreSQL 的转换设置

要编辑 Oracle 到 PostgreSQL 的转换设置，请选择 AWS SCT 中的设置，然后选择转换设置。从上面的列表中选择 Oracle，然后选择 Oracle – PostgreSQL。AWS SCT 显示了 Oracle 到 PostgreSQL 转换的所有可用设置。

AWS SCT 中的 Oracle 到 PostgreSQL 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中，为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。
- 允许 AWS SCT 将 Oracle 实体化视图转换为 PostgreSQL 表或实体化视图。对于实体化视图转换为，请选择如何转换源实体化视图。
- 在源 Oracle 代码中包含带有 PostgreSQL 不支持的参数的 `TO_CHAR`、`TO_DATE` 和 `TO_NUMBER` 函数时使用该代码。默认情况下，AWS SCT 会在转换后的代码中模拟这些参数的用法。

  当源 Oracle 代码仅包含 PostgreSQL 支持的参数时，可以使用原生 PostgreSQL `TO_CHAR`、`TO_DATE` 和 `TO_NUMBER` 函数。在这种情况下，转换后的代码运行更快。要仅包含这些参数，请选择以下值：
  - 函数 TO_CHAR() 不使用 Oracle 特定的格式化字符串
  - 函数 TO_DATE() 不使用 Oracle 特定的格式化字符串
  - 函数 TO_NUMBER() 不使用 Oracle 特定的格式化字符串
- 要解决源 Oracle 数据库在 `NUMBER` 数据类型的主键列或外键列中仅存储整数值的问题，AWS SCT 可以将这些列转换为 `BIGINT` 数据类型。这种方法可提高转换后的代码的性能。要采用这种方法，请选择将 NUMBER 主/外键列转换为 BIGINT 列。请确保源在这些列中不包含浮点值，以避免数据丢失。
- 跳过源代码中已停用的触发器和约束。为此，请选择忽略禁用的触发器和约束。
- 使用 AWS SCT 转换被称为动态 SQL 的字符串变量。数据库代码可以更改这些字符串变量的值。要确保 AWS SCT 始终转换此字符串变量的最新值，请选择转换在调用的例程中创建的动态 SQL 代码。
- 解决 PostgreSQL 版本 10 及更早版本不支持过程的问题。如果您或您的用户不熟悉在 PostgreSQL 中使用过程，AWS SCT 可以将 Oracle 过程转换为 PostgreSQL 函数。为此，请选择将过程转换为函数。
- 查看有关已发生操作项的更多信息。为此，您可以通过选择添加异常，提高迁移问题块严重性级别向扩展包中添加特定函数。然后，选择引发用户定义的异常的严重性级别。
- 使用可能包含自动生成名称的约束的源 Oracle 数据库。如果源代码使用这些名称，请确保选择使用源的原始名称转换系统生成的约束名称。如果源代码使用这些约束，但未使用其名称，请清除此选项以提高转换速度。

- 解决数据库和应用程序在不同的时区运行的问题。默认情况下，AWS SCT 在转换后的代码中模拟时区。但是，当数据库和应用程序使用相同的时区时，您不需要这种模拟。在这种情况下，选择客户端时区与服务器端时区相匹配。
- 解决源数据库和目标数据库在不同的时区运行的问题。如果是这样，模拟 SYSDATE 内置 Oracle 函数的函数会返回与源函数不同的值。要确保源函数和目标函数返回的值相同，请选择为 SYSDATE 模拟设置默认时区。
- 在转换后的代码中使用 orafce 扩展中的函数。为此，在使用 orafce 实施中，选择要使用的函数。有关 orafce 的更多信息，请参阅 GitHub 上的 orafce。

转换 Oracle 序列

AWS SCT 将序列从 Oracle 转换为 PostgreSQL。如果您使用序列维护完整性约束，请确保迁移序列的新值不会与现有值重叠。

使用源数据库中的最后一个值填充转换后的序列

1. 以 Oracle 为源打开 AWS SCT 项目。
2. 选择设置，然后选择转换设置。
3. 从上面的列表中选择 Oracle，然后选择 Oracle – PostgreSQL。AWS SCT 显示了 Oracle 到 PostgreSQL 转换的所有可用设置。
4. 选择使用源端生成的最后一个值填充转换后的序列。
5. 选择确定以保存设置并关闭转换设置对话框。

转换 Oracle ROWID

在 Oracle 数据库中，ROWID 伪列包含表行的地址。ROWID 伪列是 Oracle 所特有的，因此 AWS SCT 可以在 PostgreSQL 上将 ROWID 伪列转换为数据列。通过使用此转换，您可以保留 ROWID 信息。

当 AWS SCT 转换 ROWID 伪列时，将使用 `bigint` 数据类型创建一个数据列。如果不存在主键，AWS SCT 会将 ROWID 列设置为主键。如果存在主键，AWS SCT 将设置具有唯一约束的 ROWID 列。

如果源数据库代码包含对 ROWID 的操作（无法使用数字数据类型运行这些操作），则 AWS SCT 可以使用 `character varying` 数据类型创建数据列。

为项目的 Oracle ROWID 创建数据列

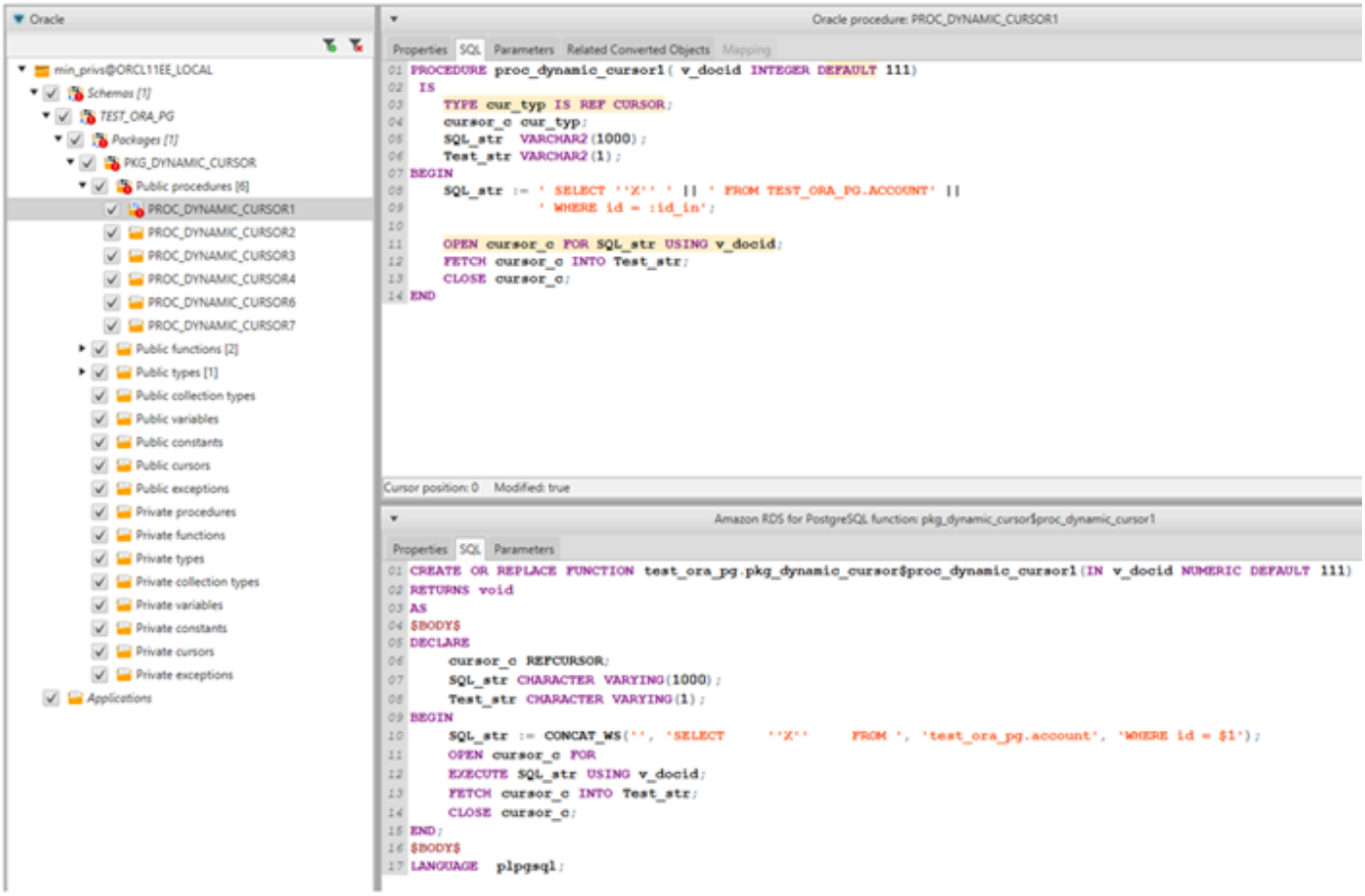
1. 以 Oracle 为源打开 AWS SCT 项目。
2. 选择设置，然后选择转换设置。
3. 从上面的列表中选择 Oracle，然后选择 Oracle – PostgreSQL。AWS SCT 显示了 Oracle 到 PostgreSQL 转换的所有可用设置。
4. 对于生成行 ID，执行以下操作之一：
   - 选择生成为身份以创建数字数据列。
   - 选择生成为字符域类型以创建字符数据列。
5. 选择确定以保存设置并关闭转换设置对话框。

转换 Oracle 动态 SQL

Oracle 提供了两种实施动态 SQL 的方法：使用 EXECUTE IMMEDIATE 语句或在 DBMS_SQL 包中调用过程。如果源 Oracle 数据库包含使用动态 SQL 的对象，请使用 AWS SCT 将 Oracle 动态 SQL 语句转换为 PostgreSQL。

将 Oracle 动态 SQL 转换为 PostgreSQL

1. 以 Oracle 为源打开 AWS SCT 项目。
2. 在 Oracle 源树视图中选择使用动态 SQL 的数据库对象。
3. 打开该对象的上下文 (右键单击) 菜单，然后选择 转换架构，并同意替换对象（如果存在）。以下屏幕截图显示了使用动态 SQL 的 Oracle 过程下面的已转换过程。



转换 Oracle 分区

AWS SCT 目前支持以下分区方法：

- 范围
- List
- 多列范围
- 哈希
- 复合（列表-列表、范围-列表、列表-范围、列表-哈希、范围-哈希、哈希-哈希）

将 Oracle 转换为 Amazon RDS for MySQL 或 Amazon Aurora MySQL

要在转换后的 MySQL 代码中模拟 Oracle 数据库函数，请使用 AWS SCT 中的 Oracle 到 MySQL 扩展包。有关扩展包的更多信息，请参阅使用 AWS SCT 扩展包。

主题

将 MySQL 用作目标数据库的权限

下面列出了将 MySQL 用作目标所需的权限：

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

如果使用版本 5.7 或更低版本的 MySQL 数据库作为目标，请授予 INVOKE LAMBDA *.* 权限，而不是 AWS_LAMBDA_ACCESS。对于 8.0 及更高版本的 MySQL 数据库，请授予 AWS_LAMBDA_ACCESS 权限。

您可以使用以下代码示例创建数据库用户并授予权限。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
```

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *your_password* 替换为安全密码。

如果使用版本 5.7 或更低版本的 MySQL 数据库作为目标，请使用 `GRANT INVOKE LAMBDA ON *.* TO 'user_name'`，而非 `GRANT AWS_LAMBDA_ACCESS TO 'user_name'`。

要使用 Amazon RDS for MySQL 或 Aurora MySQL 作为目标，请将 `lower_case_table_names` 参数设置为 1。此值意味着 MySQL 服务器在处理表、索引、触发器和数据库等对象名称的标识符时不区分大小写。如果目标实例中已开启二进制日志记录，请将 `log_bin_trust_function_creators` 参数设置为 1。在这种情况下，您无需使用 `DETERMINISTIC`、`READS SQL DATA` 或 `NO SQL` 特性创建存储函数。要配置这些参数，请创建新的数据库参数组或修改现有数据库参数组。

Oracle 到 MySQL 的转换设置

要编辑 Oracle 到 MySQL 的转换设置，请选择 AWS SCT 中的设置，然后选择转换设置。从上面的列表中选择 Oracle，然后选择 Oracle – MySQL。AWS SCT 显示了 Oracle 到 MySQL 转换的所有可用设置。

AWS SCT 中的 Oracle 到 MySQL 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中，为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 为了解决源 Oracle 数据库可以使用 ROWID 伪列但是 MySQL 不支持类似功能的问题，AWS SCT 可以在转换后的代码中模拟 ROWID 伪列。为此，请为生成行 ID？选择生成为身份 。

  如果源 Oracle 代码不使用 ROWID 伪列，请为生成行 ID？选择不生成。在这种情况下，转换后的代码运行更快。
- 在 Oracle 源代码中包含带有 MySQL 不支持的参数的 TO_CHAR、TO_DATE 和 TO_NUMBER 函数时使用该代码。默认情况下，AWS SCT 会在转换后的代码中模拟这些参数的用法。

  当源 Oracle 代码仅包含 PostgreSQL 支持的参数时，您可以使用原生 MySQL TO_CHAR、TO_DATE 和 TO_NUMBER 函数。在这种情况下，转换后的代码运行更快。要仅包含这些参数，请选择以下值：
  - 函数 TO_CHAR() 不使用 Oracle 特定的格式化字符串
  - 函数 TO_DATE() 不使用 Oracle 特定的格式化字符串
  - 函数 TO_NUMBER() 不使用 Oracle 特定的格式化字符串
- 解决数据库和应用程序在不同的时区运行的问题。默认情况下，AWS SCT 在转换后的代码中模拟时区。但是，当数据库和应用程序使用相同的时区时，您不需要这种模拟。在这种情况下，选择客户端时区与服务器端时区相匹配。

迁移注意事项

将 Oracle 转换为 RDS for MySQL 或 Aurora MySQL 时，要更改语句的运行顺序，您可以使用 GOTO 语句和标签。将跳过 GOTO 语句后的任何 PL/SQL 语句并从标签位置继续处理。可在过程、批处理或语句块中的任意位置使用 GOTO 语句和标签。GOTO 语句也可以嵌套。

MySQL 不使用 GOTO 语句。当 AWS SCT 转换包含 GOTO 语句的代码时，它将转换此语句以使用 BEGIN…END 或 LOOP…END LOOP 语句。

在下表中可以找到有关 AWS SCT 如何转换 GOTO 语句的示例。

Oracle 语句	MySQL 语句
BEGIN .... statement1; .... GOTO label1; statement2; label1:	BEGIN label1: BEGIN .... statement1; .... LEAVE label1; statement2;

Oracle 语句	MySQL 语句
Statement3; .... END	.... END; Statement3; .... END
BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	BEGIN .... statement1; .... label1: LOOP statement2; .... ITERATE label1; LEAVE label1; END LOOP; statement3; .... statement4; .... END

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

将 Oracle 中的 WITH 语句转换为 RDS for MySQL 或 Amazon Aurora MySQL

您可使用 Oracle 中的 WITH 子句（subquery_factoring）将名称（query_name）分配到子查询块。然后，您可以通过指定查询名称来引用查询中的子查询块多位置。如果子查询块不包含链接或参数（本地、过程、函数、包），则 AWS SCT 会将子句转换为视图或临时表。

将子句转换为临时表的好处是，对子查询的重复引用可能更有效。效率之所以更高，是因为可以轻松地从临时表中检索数据，而不是每次引用都需要数据。您可以通过使用额外的视图或临时表模拟。视图名称使用格式 <procedure_name>$<subselect_alias>。

您可以在下表中找到示例。

Oracle 语句	MySQL 语句

CREATE PROCEDURE
 TEST_ORA_PG.P_WITH_SELECT_V
ARIABLE_01
     (p_state IN NUMBER)
AS
  l_dept_id NUMBER := 1;
BEGIN
FOR cur IN
           (WITH dept_empl(id, name,
 surname,
              lastname, state, dept_id)
              AS
                  (
                    SELECT id, name,
 surname,
                     lastname, state,
 dept_id
                      FROM test_ora_
pg.dept_employees
                     WHERE state =
 p_state AND
                       dept_id =
 l_dept_id)
            SELECT id,state
              FROM dept_empl
            ORDER BY id) LOOP
  NULL;
END LOOP;
	CREATE PROCEDURE test_ora_pg.P_WITH
	_SELECT_VARIABLE_01(IN par_P_STATE
	 DOUBLE)
	BEGIN
	    DECLARE var_l_dept_id DOUBLE
	 DEFAULT 1;
	    DECLARE var$id VARCHAR (8000);
	    DECLARE var$state VARCHAR (8000);
	    DECLARE done INT DEFAULT FALSE;
	    DECLARE cur CURSOR FOR SELECT
	        ID, STATE
	        FROM (SELECT
	            ID, NAME, SURNAME,
	 LASTNAME, STATE, DEPT_ID
	            FROM TEST_ORA_PG.DEPT_E
	MPLOYEES
	            WHERE STATE = par_p_sta
	te AND DEPT_ID = var_l_dept_id) AS
	 dept_empl
	        ORDER BY ID;
	    DECLARE CONTINUE HANDLER FOR NOT
	 FOUND
	        SET done := TRUE;
	    OPEN cur;
	
	    read_label:
	    LOOP

Oracle 语句	MySQL 语句

	FETCH cur INTO varid, varstate;
	
	IF done THEN
	LEAVE read_label;
	END IF;
	
	BEGIN
	END;
	END LOOP;
	CLOSE cur;
	END;

Oracle 语句	MySQL 语句
CREATE PROCEDURE TEST_ORA_PG.P_WITH_SELECT_R EGULAR_MULT_01 AS BEGIN FOR cur IN (WITH dept_empl AS (SELECT id, name, surname, lastname, state, dept_id FROM test_ora_pg.dept_employees WHERE state = 1), dept AS (SELECT id deptid, parent_id, name deptname FROM test_ora_ pg.department) SELECT dept_empl .*,dept.* FROM dept_empl, dept WHERE dept_empl .dept_id = dept.deptid) LOOP NULL; END LOOP;	CREATE VIEW TEST_ORA_PG.`P_WIT H_SELECT_REGULAR_MULT_01$dept_empl  `(id, name, surname, lastname, state, dept_id) AS (SELECT id, name, surname, lastname, state, dept_id FROM test_ora_pg.dept_employees WHERE state = 1); CREATE VIEW TEST_ORA_PG.`P_WIT H_SELECT_REGULAR_MULT_01$dept  `(deptid, parent_id,deptname) AS (SELECT id deptid, parent_id, name deptname FROM test_ora_pg.department); CREATE PROCEDURE test_ora_pg.P_WITH _SELECT_REGULAR_MULT_01() BEGIN DECLARE var$ID DOUBLE; DECLARE var$NAME VARCHAR (30); DECLARE var$SURNAME VARCHAR (30); DECLARE var$LASTNAME VARCHAR (30); DECLARE var$STATE DOUBLE; DECLARE var$DEPT_ID DOUBLE; DECLARE var$deptid DOUBLE; DECLARE var$PARENT_ID DOUBLE; DECLARE var$deptname VARCHAR (200); DECLARE done INT DEFAULT FALSE; DECLARE cur CURSOR FOR SELECT dept_empl.*, dept.* FROM TEST_ORA_PG.`P_WIT H_SELECT_REGULAR_MULT_01$dept_empl   ` AS dept_empl, TEST_ORA_PG.`P_WIT H_SELECT_REGULAR_MULT_01$dept   ` AS dept

Oracle 语句	MySQL 语句

	WHERE dept_empl.DEPT_ID = dept.DEPTID;
	DECLARE CONTINUE HANDLER FOR NOT FOUND
	SET done := TRUE;
	OPEN cur;
	
	read_label:
	LOOP
	FETCH cur INTO varID, varNAME, var$SURNAME,
	var$LASTNAME, var$STATE, var$DEPT_ID, var$deptid,
	var$PARENT_ID, var$deptname;
	
	IF done THEN
	LEAVE read_label;
	END IF;
	
	BEGIN
	END;
	END LOOP;
	CLOSE cur;
	END;
	
	call test_ora_pg.P_WITH_SELECT_REGULAR_MULT_01()

Oracle 语句	MySQL 语句
CREATE PROCEDURE TEST_ORA_PG.P_WITH_SELECT_V AR_CROSS_02(p_state IN NUMBER) AS l_dept_id NUMBER := 10; BEGIN FOR cur IN (WITH emp AS (SELECT id, name, surname, lastname, state, dept_id FROM test_ora_ pg.dept_employees WHERE dept_id > 10), active_emp AS (SELECT id FROM emp WHERE emp.state = p_state) SELECT * FROM active_emp) LOOP NULL; END LOOP; END;	CREATE VIEW TEST_ORA_PG.`P_WIT H_SELECT_VAR_CROSS_01$emp     `(id, name, surname, lastname, state, dept_id) AS (SELECT id, name, surname, lastname, state, dept_id FROM TEST_ORA_PG.DEPT_EMPLOYEES WHERE DEPT_ID > 10); CREATE PROCEDURE test_ora_pg.P_WITH_SELECT_V AR_CROSS_02(IN par_P_STATE DOUBLE) BEGIN DECLARE var_l_dept_id DOUBLE DEFAULT 10; DECLARE var$ID DOUBLE; DECLARE done INT DEFAULT FALSE; DECLARE cur CURSOR FOR SELECT * FROM (SELECT ID FROM TEST_ORA_ PG. `P_WITH_S ELECT_VAR_CROSS_01$emp` AS emp WHERE emp.STATE = par_p_state) AS active_emp; DECLARE CONTINUE HANDLER FOR NOT FOUND SET done := TRUE; OPEN cur; read_label:

Oracle 语句	MySQL 语句

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

将 Oracle 转换为 Amazon RDS for Oracle

将 Oracle 架构和代码迁移到 Amazon RDS for Oracle 时要考虑的一些事项：

- AWS SCT 可以将目录对象添加到对象树中。目录对象是逻辑结构，每个结构代表服务器文件系统上的物理目录。可以使用包含包（如 DBMS_LOB、UTL_FILE、DBMS_FILE_TRANSFER、DATAPUMP 实用程序等）的目录对象。
- AWS SCT 支持将 Oracle 表空间转换为 Amazon RDS for Oracle 数据库实例。Oracle 将数据存储在表空间中 (按逻辑) 和与相应表空间关联的数据文件中 (以物理方式)。在 Oracle 中，可以创建包含数据文件名的表空间。Amazon RDS 仅支持数据文件、日志文件和控制文件的 Oracle Managed Files（OMF）。AWS SCT 在转换过程中创建所需的数据文件。
- AWS SCT 可以转换服务器级别的角色和权限。Oracle 数据库引擎采用了基于角色的安全机制。角色是可授予用户或从用户撤消的特权的集合。Amazon RDS 中名为 DBA 的预定义角色一般允许 Oracle 数据库引擎上的所有管理特权。对于使用 Oracle 引擎的 Amazon RDS 数据库实例，DBA 角色没有以下权限：
  - 更改数据库
  - 更改系统
  - 创建任何目录
  - 授予任何权限
  - 授予任何角色
  - 创建外部任务

您可以向 Amazon RDS for Oracle 用户角色授予所有其他权限，包括高级筛选和列权限。

- AWS SCT 支持将 Oracle 作业转换为可在 Amazon RDS for Oracle 中运行的作业。转换存在一些限制，其中包括：
  - 不支持可执行文件作业。
  - 不支持使用 ANYDATA 数据类型作为参数的计划作业。
- Oracle Real Application Clusters (RAC) One Node 是在 Oracle Database 11g Release 2 中引入的 Oracle Database Enterprise Edition 的选件。Amazon RDS for Oracle 不支持 RAC 功能。为了实现高可用性，可使用 Amazon RDS 多可用区。

  在多可用区部署中，Amazon RDS 会自动在不同可用区中配置和维护一个同步备用副本。主数据库实例可以跨可用区同步复制到备用副本。此功能提供数据冗余，消除 I/O 冻结，并在系统备份期间将延迟峰值降至最小。
- Oracle Spatial 提供了一个 SQL 架构和一些功能，可帮助在 Oracle 数据库中存储、检索、更新和查询空间数据集合。Oracle Locator 提供支持基于 Internet 和无线服务的应用程序与基于合作伙伴的 GIS 解决方案通常所需的功能。Oracle Locator 是 Oracle Spatial 的有限子集。

  要使用 Oracle Spatial 和 Oracle Locator 功能，可以将 SPATIAL 选项或 LOCATOR 选项（互斥）添加到数据库实例的选项组。

  在 Amazon RDS for Oracle 数据库实例上使用 Oracle Spatial 和 Oracle Locator 有一些先决条件：
  - 该实例应使用 Oracle 企业版 12.1.0.2.v6 版或更高版本，或者 11.2.0.4.v10 版或更高版本。
  - 该实例应在 Virtual Private Cloud (VPC) 内。
  - 该实例应该是可支持 Oracle 功能的数据库实例类。例如，db.m1.small、db.t1.micro、db.t2.micro 或 db.t2.small 数据库实例类不支持 Oracle Spatial。有关更多信息，请参阅 Oracle 的数据库实例类支持。
  - 实例必须已启用“Auto Minor Version Upgrade (自动次要版本升级)”选项。如果存在 CVSS 评分为 9+ 的安全漏洞或其他公布的安全漏洞，则 Amazon RDS 会将数据库实例更新为最新的 Oracle PSU。有关更多信息，请参阅

    适用于 Oracle 数据库实例的设置。
  - 如果数据库实例为 11.2.0.4.v10 版或更高版本，则必须安装 XMLDB 选项。有关更多信息，请参阅

    Oracle XML 数据库。

  - 您应该拥有 Oracle 中的 Oracle Spatial 许可证。有关更多信息，请参阅 Oracle 文档中的 Oracle Spatial 和图表。
- Data Guard 随 Oracle Database Enterprise Edition 提供。为了实现高可用性，可使用 Amazon RDS 多可用区功能。

  在多可用区部署中，Amazon RDS 会自动在不同可用区中配置和维护一个同步备用副本。主数据库实例可以跨可用区同步复制到备用副本。此功能提供数据冗余，消除 I/O 冻结，并在系统备份期间将延迟峰值降至最小。
- AWS SCT 支持在迁移至 Amazon RDS for Oracle 时转换 Oracle DBMS_SCHEDULER 对象。AWS SCT 评估报告会指明是否可以转换计划对象。有关将计划对象用于 Amazon RDS 的详细信息，请参阅 Amazon RDS 文档。
- 对于 Oracle 到 Amazon RDS for Oracle 转换，支持数据库链接。数据库链接是一个数据库中的架构对象，允许您访问另一个数据库上的对象。另一个数据库不需要是 Oracle 数据库。但是，要访问非 Oracle 数据库，您必须使用 Oracle 异构服务。

  一旦您创建数据库链接，便可以使用 SQL 语句中的链接来引用其他数据库中的表、视图和 PL/SQL 对象。要使用数据库链接，请将 `@dblink` 附加到表、视图或 PL/SQL 对象名称。您可以使用 SELECT 语句查询其他数据库中的表或视图。有关使用 Oracle 数据库链接的更多信息，请参阅 Oracle 文档。

  有关将数据库链接与 Amazon RDS 一起使用的更多信息，请参阅 Amazon RDS 文档。
- AWS SCT 评估报告提供了转换的服务器指标。这些有关 Oracle 实例的指标包括：
  - 目标数据库实例的计算和内存容量。
  - 不支持的 Oracle 功能。例如，Amazon RDS 不支持的 Real Application Clusters。
  - 磁盘读写负载
  - 平均总磁盘吞吐量
  - 服务器信息，如服务器名称、操作系统、主机名和字符集。

将 Oracle 用作目标的权限

要迁移到 Amazon RDS for Oracle，请创建特权数据库用户。您可以使用以下代码示例：

```
CREATE USER user_name IDENTIFIED BY your_password;

-- System privileges
GRANT DROP ANY CUBE BUILD PROCESS TO user_name;
```

```
GRANT ALTER ANY CUBE TO user_name;
GRANT CREATE ANY CUBE DIMENSION TO user_name;
GRANT CREATE ANY ASSEMBLY TO user_name;
GRANT ALTER ANY RULE TO user_name;
GRANT SELECT ANY DICTIONARY TO user_name;
GRANT ALTER ANY DIMENSION TO user_name;
GRANT CREATE ANY DIMENSION TO user_name;
GRANT ALTER ANY TYPE TO user_name;
GRANT DROP ANY TRIGGER TO user_name;
GRANT CREATE ANY VIEW TO user_name;
GRANT ALTER ANY CUBE BUILD PROCESS TO user_name;
GRANT CREATE ANY CREDENTIAL TO user_name;
GRANT DROP ANY CUBE DIMENSION TO user_name;
GRANT DROP ANY ASSEMBLY TO user_name;
GRANT DROP ANY PROCEDURE TO user_name;
GRANT ALTER ANY PROCEDURE TO user_name;
GRANT ALTER ANY SQL TRANSLATION PROFILE TO user_name;
GRANT DROP ANY MEASURE FOLDER TO user_name;
GRANT CREATE ANY MEASURE FOLDER TO user_name;
GRANT DROP ANY CUBE TO user_name;
GRANT DROP ANY MINING MODEL TO user_name;
GRANT CREATE ANY MINING MODEL TO user_name;
GRANT DROP ANY EDITION TO user_name;
GRANT CREATE ANY EVALUATION CONTEXT TO user_name;
GRANT DROP ANY DIMENSION TO user_name;
GRANT ALTER ANY INDEXTYPE TO user_name;
GRANT DROP ANY TYPE TO user_name;
GRANT CREATE ANY PROCEDURE TO user_name;
GRANT CREATE ANY SQL TRANSLATION PROFILE TO user_name;
GRANT CREATE ANY CUBE TO user_name;
GRANT COMMENT ANY MINING MODEL TO user_name;
GRANT ALTER ANY MINING MODEL TO user_name;
GRANT DROP ANY SQL PROFILE TO user_name;
GRANT CREATE ANY JOB TO user_name;
GRANT DROP ANY EVALUATION CONTEXT TO user_name;
GRANT ALTER ANY EVALUATION CONTEXT TO user_name;
GRANT CREATE ANY INDEXTYPE TO user_name;
GRANT CREATE ANY OPERATOR TO user_name;
GRANT CREATE ANY TRIGGER TO user_name;
GRANT DROP ANY ROLE TO user_name;
GRANT DROP ANY SEQUENCE TO user_name;
GRANT DROP ANY CLUSTER TO user_name;
GRANT DROP ANY SQL TRANSLATION PROFILE TO user_name;
GRANT ALTER ANY ASSEMBLY TO user_name;
```

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

```
GRANT ALTER ANY TRIGGER TO user_name;
GRANT CREATE ANY SEQUENCE TO user_name;
GRANT DROP ANY INDEX TO user_name;
GRANT CREATE ANY INDEX TO user_name;
GRANT DROP ANY TABLE TO user_name;
GRANT SELECT_CATALOG_ROLE TO user_name;
GRANT SELECT ANY SEQUENCE TO user_name;

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


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

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

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

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

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

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *your_password* 替换为安全密码。

将 Oracle 转换为 Amazon RDS for Oracle 的限制

将 Oracle 架构和代码迁移到 Amazon RDS for Oracle 时应考虑的一些限制：

- Amazon RDS 中名为 DBA 的预定义角色一般允许 Oracle 数据库引擎上的所有管理特权。对于使用 Oracle 引擎的 Amazon RDS 数据库实例，DBA 角色没有以下权限：
  - 更改数据库
  - 更改系统
  - 创建任何目录
  - 授予任何权限
  - 授予任何角色
  - 创建外部任务

  可以向 Oracle RDS 用户角色授予所有其他特权。
- Amazon RDS for Oracle 支持传统审核、使用 DBMS_FGA 包的精细审核以及 Oracle 的统一审核。
- Amazon RDS for Oracle 不支持更改数据捕获 (CDC)。要在数据库迁移期间和迁移之后执行更改数据捕获，请使用 AWS Database Migration Service。

将 PostgreSQL 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、数据库代码对象和应用程序代码从 PostgreSQL 转换到以下目标：

- Amazon RDS for MySQL
- Amazon Aurora MySQL 兼容版
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版

有关详细信息，请参阅以下章节：

主题

将 PostgreSQL 用作源数据库的权限

下面列出了将 PostgreSQL 用作源所需的权限：

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

连接到作为源的 PostgreSQL

使用 AWS Schema Conversion Tool 按照以下过程连接到 PostgreSQL 源数据库。

连接到 PostgreSQL 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 PostgreSQL，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，请选择密钥的名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅 使用 AWS Secrets Manager。
   - 要手动输入 PostgreSQL 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。 您可以使用 IPv6 地址协议连接到源 PostgreSQL 数据库。为此，请确保使用方括号输入 IP 地址，如以下示例所示。 `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
服务器端口	输入用于连接到源数据库服务器的端口。

参数	操作
数据库。	输入 PostgreSQL 数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层（SSL）连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 验证服务器证书：选择此选项以使用信任存储验证服务器证书。 • 信任存储：包含证书的信任存储的位置。要使此位置显示在全局设置部分，请务必将其添加。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项可让您存储数据库密码并在不需要输入密码的情况下快速连接到数据库。
PostgreSQL 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 MySQL 用作目标数据库的权限

从 PostgreSQL 迁移时，MySQL 作为目标所需的权限如下：

- CREATE ON *.*

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

您可以使用以下代码示例创建数据库用户并授予权限。

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *your_password* 替换为安全密码。

要使用 Amazon RDS for MySQL 或 Aurora MySQL 作为目标，请将 `lower_case_table_names` 参数设置为 1。此值意味着 MySQL 服务器在处理表、索引、触发器和数据库等对象名称的标识符时不区分大小写。如果目标实例中已开启二进制日志记录，请将 `log_bin_trust_function_creators` 参数设置为 1。在这种情况下，您无需使用 `DETERMINISTIC`、`READS SQL DATA` 或 `NO SQL` 特性创建存储函数。要配置这些参数，请创建新的数据库参数组或修改现有数据库参数组。

使用 SAP ASE（Sybase ASE）作为 AWS SCT 的源

您可以使用 AWS SCT 将架构、数据库代码对象和应用程序代码从 SAP (Sybase) Adaptive Server Enterprise (ASE) 转换为以下目标：

- Amazon RDS for MySQL
- Amazon Aurora MySQL 兼容版
- Amazon RDS for MariaDB
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版

有关详细信息，请参阅以下章节：

主题

将 SAP ASE 用作源数据库的权限

要将 SAP ASE 数据库作为源，您需要创建数据库用户并授予权限。为此，请执行以下步骤：

创建和配置数据库用户

1. 连接到源数据库。
2. 使用以下命令创建数据库用户。提供新用户密码。

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

3. 为要迁移的每个数据库授予以下权限。

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

连接到作为源的 SAP ASE (Sybase)

使用 AWS Schema Conversion Tool 按照以下过程连接到 SAP ASE 源数据库。

连接到 SAP ASE 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 SAP ASE，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，请选择密钥的名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。

- 要手动输入 SAP ASE 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
数据库。	输入 SAP ASE 数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 **Note** 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层 (SSL) 连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 验证服务器证书：选择此选项以使用信任存储验证服务器证书。 • 信任存储：包含证书的信任存储的位置。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项可让您存储数据库密码并在不需要输入密码的情况下快速连接到数据库。
SAP ASE 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 MySQL 用作目标数据库的权限

下面列出了将 MySQL 用作目标所需的权限：

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

您可以使用以下代码示例创建数据库用户并授予权限。

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

```
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_SAPASE_EXT.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_SAPASE_EXT.* TO 'user_name';
```

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *your_password* 替换为安全密码。

要使用 Amazon RDS for MySQL 或 Aurora MySQL 作为目标，请将 lower_case_table_names 参数设置为 1。此值意味着 MySQL 服务器在处理表、索引、触发器和数据库等对象名称的标识符时不区分大小写。如果目标实例中已开启二进制日志记录，请将 log_bin_trust_function_creators 参数设置为 1。在这种情况下，您无需使用 DETERMINISTIC、READS SQL DATA 或 NO SQL 特性创建存储函数。要配置这些参数，请创建新的数据库参数组或修改现有数据库参数组。

SAP ASE 到 MySQL 的转换设置

要编辑 SAP ASE 到 MySQL 的转换设置，请选择设置，然后选择转换设置。从上面的列表中选择 SAP ASE，然后选择 SAP ASE – MySQL 或 SAP ASE – Amazon Aurora（兼容 MySQL）。AWS SCT 显示 SAP ASE 到 PostgreSQL 转换的所有可用设置。

AWS SCT 中的 SAP ASE 到 MySQL 转换设置包括以下选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中，为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。
- 在转换后的代码中使用源数据库对象的确切名称。

  默认情况下，AWS SCT 将数据库对象、变量和参数的名称转换为小写。要保持这些名称的原始大小写，请选择源数据库对象名称区分大小写。如果源 SAP ASE 数据库服务器中使用区分大小写的对象名称，请选择此选项。

将 PostgreSQL 用作目标数据库的权限

要使用 PostgreSQL 作为目标，AWS SCT 需要 CREATE ON DATABASE 权限。请确保为每个目标 PostgreSQL 数据库授予此权限。

要使用转换后的公共同义词，请将数据库的默认搜索路径更改为 "$user", public_synonyms, public。

您可以使用以下代码示例创建数据库用户并授予权限。

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *db_name* 替换为目标数据库名称。最后，将 *your_password* 替换为安全密码。

在 PostgreSQL 中，只有架构所有者或 superuser 才能删除架构。即使架构的所有者并不拥有架构的某些对象，该所有者也可以删除该架构及其包含的所有对象。

使用不同的用户转换不同的架构并将其应用到目标数据库时，若 AWS SCT 无法删除架构，您可能会收到一条错误消息。要避免出现此错误消息，请使用 superuser 角色。

SAP ASE 到 PostgreSQL 的转换设置

要编辑 SAP ASE 到 PostgreSQL 的转换设置，请选择设置，然后选择转换设置。从上面的列表中选择 SAP ASE，然后选择 SAP ASE – PostgreSQL 或 SAP ASE – Amazon Aurora（兼容 PostgreSQL）。AWS SCT 显示 SAP ASE 到 PostgreSQL 转换的所有可用设置。

AWS SCT 中 SAP ASE 到 PostgreSQL 的转换设置包括以下选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中，为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。
- 定义要在转换后的代码中用于架构名称的模板。对于架构名称生成模板，选择以下选项之一：
  - <source_db>：使用 SAP ASE 数据库名称作为 PostgreSQL 中的架构名称。
  - <source_schema>：使用 SAP ASE 架构名称作为 PostgreSQL 中的架构名称。
  - <source_db>_<schema>：使用 SAP ASE 数据库名称和架构名称的组合作为 PostgreSQL 中的架构名称。
- 在转换后的代码中使用源数据库对象的确切名称。

默认情况下，AWS SCT 将数据库对象、变量和参数的名称转换为小写。要保持这些名称的原始大小写，请选择源数据库对象名称区分大小写。如果源 SAP ASE 数据库服务器中使用区分大小写的对象名称，请选择此选项。

对于区分大小写的操作，AWS SCT 可以避免将数据库对象名称转换为小写。为此，请选择避免将区分大小写的操作转换为小写。

- 允许在 SAP ASE 的不同表中使用同名索引。

  在 PostgreSQL 中，架构中使用的所有索引名称都必须是唯一的。要确保 AWS SCT 为所有索引生成唯一的名称，请选择为索引生成唯一名称。

将 Microsoft SQL Server 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、数据库代码对象和应用程序代码从 SQL Server 转换到以下目标：

- Amazon RDS for MySQL
- Amazon Aurora MySQL 兼容版
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL 兼容版
- Amazon RDS for SQL Server
- Amazon RDS for MariaDB

> **Note**
>
> AWS SCT 不支持将 Amazon RDS for SQL Server 作为源。

您可以使用 AWS SCT 创建评估报告，用于评估将架构、数据库代码对象和应用程序代码从 SQL Server 迁移到适用于 Aurora PostgreSQL 的 Babelfish 的结果，如下所述。

主题

- 将 SQL Server 转换为 PostgreSQL
- 将 SQL Server 转换为 Amazon RDS for SQL Server

将 Microsoft SQL Server 用作源的权限

下面列出了将 Microsoft SQL Server 用作源所需的权限：

- VIEW DEFINITION
- VIEW DATABASE STATE

VIEW DEFINITION 权限允许具有公共访问权限的用户查看对象定义。AWS SCT 使用 VIEW DATABASE STATE 权限检查 SQL Server 企业版的功能。

对要转换其架构的每个数据库重复这种授权。

此外，授予针对 master 数据库的以下权限：

- VIEW SERVER STATE
- VIEW ANY DEFINITION

AWS SCT 使用 VIEW SERVER STATE 权限收集服务器设置和配置。确保授予 VIEW ANY DEFINITION 权限以便查看端点。

要读取有关 Microsoft Analysis Services 的信息，请在 master 数据库上运行以下命令。

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

在上述示例中，将 *<user_name>* 占位符替换为您之前授予权限的用户名。

要阅读有关 SQL Server Agent 的信息，请将用户添加到 SQLAgentUser 角色中。在 msdb 数据库上运行以下命令。

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

在上述示例中，将 *<SQLAgentRole>* 占位符替换为 SQL Server Agent 角色的名称。然后将 *<user_name>* 占位符替换为您之前授予权限的用户名。有关更多信息，请参阅《Amazon RDS 用户指南》中的将用户添加到 SQLAgentUser 角色。

要检测日志传输，请授予针对 msdb 数据库的 SELECT on dbo.log_shipping_primary_databases 权限。

要使用 DDL 复制的通知方法，请授予针对源数据库的 RECEIVE ON *<schema_name>*.*<queue_name>* 权限。在本例中，将 *<schema_name>* 占位符替换为数据库的架构名称。然后，将 *<queue_name>* 占位符替换为队列表的名称。

当将 Microsoft SQL Server 用作源时使用 Windows 身份验证

如果您的应用程序在基于 Windows 的 Intranet 上运行，您可能可以使用 Windows 身份验证进行数据库访问。Windows 身份验证使用在操作系统线程上建立的当前 Windows 身份来访问 SQL Server 数据库。然后，您可以将 Windows 身份映射到 SQL Server 数据库和权限。要使用 Windows 身份验证连接到 SQL Server，您必须指定您的应用程序正在使用的 Windows 身份。您还必须向 Windows 身份授予访问 SQL Server 数据库的权限。

SQL Server 有两种访问模式：Windows 身份验证模式和混合模式。Windows 身份验证模式启用 Windows 身份验证，禁用 SQL Server 身份验证。混合模式启用 Windows 身份验证和 SQL Server 身份验证。Windows 身份验证是始终可用的，无法禁用。有关 Windows 身份验证的更多信息，请参阅 Microsoft Windows 文档。

在 TEST_DB 中创建用户的可能示例如下所示。

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

将 Windows 身份验证与 JDBC 连接配合使用

在非 Windows 操作系统上使用 JDBC 驱动程序时，驱动程序不支持 Windows 身份验证。当从非 Windows 操作系统连接到 SQL Server 时，不会自动指定 Windows 身份验证凭证，例如用户名和密码。在这种情况下，应用程序必须改而使用 SQL Server 身份验证。

在 JDBC 连接字符串中，必须指定参数 integratedSecurity ，才能使用 Windows 身份验证进行连接。JDBC 驱动程序通过 integratedSecurity 连接字符串参数在 Windows 操作系统上支持集成 Windows 身份验证。

使用集成身份验证

1. 安装 JDBC 驱动程序。

2. 将 sqljdbc_auth.dll 文件复制到装有 JDBC 驱动程序的计算机上的 Windows 系统路径上的目录中。

   sqljdbc_auth.dll 文件安装在以下位置：

   <安装目录>\sqljdbc_<版本>\<语言>\auth\

当您尝试使用 Windows 身份验证建立与 SQL Server 数据库的连接时，您可能会收到以下错误：此驱动程序未针对集成身份验证进行配置。通过执行以下操作可以解决此问题：

- 声明两个指向 JDBC 的安装路径的变量：

  variable name: SQLJDBC_HOME; variable value: D:\lib\JDBC4.1\enu（sqljdbc4.jar 所在的位置）；

  variable name: SQLJDBC_AUTH_HOME; variable value: D\lib\JDBC4.1\enu\auth\x86（如果您运行的是 32 位操作系统）或 D\lib\JDBC4.1\enu\auth\x64（如果您运行的是 64 位操作系统）。这是 sqljdbc_auth.dll 所在的位置。
- 将 sqljdbc_auth.dll 复制到 JDK/JRE 正在其中运行的文件夹。您可以复制到 lib 文件夹、bin 文件夹等。例如，您可以复制到以下文件夹。

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

- 确保 JDBC 库文件夹中只有 SQLJDBC4.jar 文件。从该文件夹中删除所有其他 sqljdbc*.jar 文件（或将这些文件复制到其他文件夹）。如果要将驱动程序条件到程序，请确保只添加 SQLJDBC4.jar 以作为驱动程序使用。
- 在您的应用程序所在的文件夹中复制 sqljdbc_auth.dll 文件。

> **Note**
>
> 如果您运行的是 32 位 Java 虚拟机 (JVM)，请使用 x86 文件夹中的 sqljdbc_auth.dll 文件，即使操作系统是 x64 版本也是如此。如果您是在 x64 处理器上运行 64 位 JVM ，请使用 x64 文件夹中的 sqljdbc_auth.dll 文件。

当您连接到 SQL Server 数据库时，您可以为身份验证选项选择 Windows 身份验证或 SQL Server 身份验证。

连接到作为源的 SQL Server

使用 AWS Schema Conversion Tool 按照以下过程连接到 Microsoft SQL Server 源数据库。

连接到 Microsoft SQL Server 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Microsoft SQL Server，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，请选择密钥的名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅 使用 AWS Secrets Manager。
   - 要手动输入 Microsoft SQL Server 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名服务 (DNS) 名称或 IP 地址。 您可以使用 IPv6 地址协议连接到源 SQL Server 数据库。为此，请确保使用方括号输入 IP 地址，如以下示例所示。 `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
服务器端口	输入用于连接到源数据库服务器的端口。
实例名称	输入 SQL Server 数据库的实例名称。要找到实例名称，请在您的 SQL Server 数据库上运行查询 `SELECT @@servername;` 。

参数	操作
身份验证	从 Windows 身份验证和 SQL Server 身份验证中选择身份验证类型。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层 (SSL) 连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 信任服务器证书：选择此选项以信任服务器证书。 • 信任存储：包含证书的信任存储的位置。要使此位置显示在全局设置部分，请务必将其添加。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项可让您存储数据库密码并在不需要输入密码的情况下快速连接到数据库。
SQL Server 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。
Windows 身份验证库	输入 `sqljdbc_auth.dll` 文件的路径。默认情况下，此文件安装在以下位置： *<installation directory of the JDBC driver>*sqljdbc_*<version>* *<language>* \auth\

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

将 SQL Server 转换为 MySQL

要在转换后的 MySQL 代码中模拟 Microsoft SQL Server 数据库函数，请使用 AWS SCT 中的 SQL Server 到 MySQL 扩展包。有关扩展包的更多信息，请参阅使用 AWS SCT 扩展包。

主题

将 MySQL 用作目标数据库的权限

下面列出了将 MySQL 用作目标所需的权限：

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

您可以使用以下代码示例创建数据库用户并授予权限。

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
```

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *your_password* 替换为安全密码。

如果将 MySQL 数据库版本 5.7 或更低版本作为目标，请运行以下命令。8.0 及更高版本的 MySQL 数据库不建议使用此命令。

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

要使用 Amazon RDS for MySQL 或 Aurora MySQL 作为目标，请将 `lower_case_table_names` 参数设置为 1。此值意味着 MySQL 服务器在处理表、索引、触发器和数据库等对象名称的标识符时不区分大小写。如果目标实例中已开启二进制日志记录，请将 `log_bin_trust_function_creators` 参数设置为 1。在这种情况下，您无需使用 `DETERMINISTIC`、`READS SQL DATA` 或 `NO SQL` 特性创建存储函数。要配置这些参数，请创建新的数据库参数组或修改现有数据库参数组。

SQL Server 到 MySQL 的转换设置

要编辑 SQL Server 到 MySQL 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 SQL Server，然后选择 SQL Server – MySQL。AWS SCT 显示 SQL Server 到 MySQL 转换的所有可用设置。

AWS SCT 中的 SQL Server 到 MySQL 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中，为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 允许源 SQL Server 数据库将 EXEC 的输出存储在表中。AWS SCT 创建临时表和模拟此功能的附加过程。要使用此模拟，请选择创建额外的例程处理开放数据集。

迁移注意事项

将 SQL Server 架构迁移到 MySQL 时的注意事项：

- MySQL 不支持 MERGE 语句。但是，AWS SCT 可以在转换过程中使用 INSERT ON DUPLICATE KEY 子句和 UPDATE FROM and DELETE FROM 语句来模拟 MERGE 语句。

  对于使用 INSERT ON DUPLICATE KEY 的正确模拟，请确保目标 MySQL 数据库上存在唯一约束或主键。

- 可以使用一个 GOTO 语句和一个标签更改语句的运行顺序。将跳过接在 GOTO 语句后的任何 Transact-SQL 语句并且处理将在标签处继续。可在过程、批处理或语句块中的任意位置使用 GOTO 语句和标签。您也可以嵌套 GOTO 语句。

  MySQL 不使用 GOTO 语句。当 AWS SCT 转换包含 GOTO 语句的代码时，它将转换此语句以使用 BEGIN…END 或 LOOP…END LOOP 语句。在下表中可以找到有关 AWS SCT 如何转换 GOTO 语句的示例。

SQL Server 语句	MySQL 语句

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

SQL Server 语句	MySQL 语句
BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	BEGIN .... statement1; .... label1: LOOP statement2; .... ITERATE label1; LEAVE label1; END LOOP; statement3; .... statement4; .... END

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

- MySQL 不支持多语句表值函数。AWS SCT 会在转换过程中通过创建临时表并重写使用这些临时表的语句，来模拟表值函数。

将 SQL Server 转换为 PostgreSQL

你可以在 AWS SCT 中使用 SQL Server 到 PostgreSQL 的扩展包。此扩展包在转换后的 PostgreSQL 代码中模拟 SQL Server 数据库函数。使用 SQL Server 到 PostgreSQL 扩展包模拟 SQL Server Agent 和 SQL Server 数据库邮件。有关扩展包的更多信息，请参阅使用 AWS SCT 扩展包。

主题

将 PostgreSQL 用作目标数据库的权限

要使用 PostgreSQL 作为目标，AWS SCT 需要 `CREATE ON DATABASE` 权限。请确保为每个目标 PostgreSQL 数据库授予此权限。

要使用转换后的公共同义词，请将数据库的默认搜索路径更改为 `"$user", public_synonyms, public`。

您可以使用以下代码示例创建数据库用户并授予权限。

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

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *db_name* 替换为目标数据库名称。最后，将 *your_password* 替换为安全密码。

在 PostgreSQL 中，只有架构所有者或 `superuser` 才能删除架构。即使架构的所有者并不拥有架构的某些对象，该所有者也可以删除该架构及其包含的所有对象。

使用不同的用户转换不同的架构并将其应用到目标数据库时，若 AWS SCT 无法删除架构，您可能会收到一条错误消息。要避免出现此错误消息，请使用 `superuser` 角色。

SQL Server 到 PostgreSQL 的转换设置

要编辑 SQL Server 到 PostgreSQL 的转换设置，请选择设置，然后选择转换设置。从上方的列表中选择 SQL Server，然后选择 SQL Server – PostgreSQL。AWS SCT 显示 SQL Server 到 PostgreSQL 转换的所有可用设置。

AWS SCT 中的 SQL Server 到 PostgreSQL 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中，为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 允许在 SQL Server 的不同表中使用同名索引。

  在 PostgreSQL 中，架构中使用的所有索引名称都必须是唯一的。要确保AWS SCT为所有索引生成唯一的名称，请选择为索引生成唯一名称。

- 将 SQL Server 过程转换为 PostgreSQL 函数。

  PostgreSQL 版本 10 及更早版本不支持过程。对于不熟悉在 PostgreSQL 中使用过程的客户，AWS SCT 可以将过程转换为函数。为此，请选择将过程转换为函数。

- 在表中模拟 `EXEC` 的输出。

  源 SQL Server 数据库可以将 `EXEC` 的输出存储在表中。AWS SCT 创建临时表和模拟此功能的附加过程。要使用此模拟，请选择创建额外的例程处理开放数据集。

- 定义要在转换后的代码中用于架构名称的模板。对于架构名称生成模板，选择以下选项之一：
  - <source_db>：使用 SQL Server 数据库名称作为 PostgreSQL 中的架构名称。
  - <source_schema>：使用 SQL Server 架构名称作为 PostgreSQL 中的架构名称。
  - <source_db>_<schema>：使用 SQL Server 数据库名称和架构名称的组合作为 PostgreSQL 中的架构名称。
- 保持源对象名称的字母大小写。

  要避免将对象名称转换为小写，请选择避免将区分大小写的操作转换为小写。仅当您在目标数据库中启用区分大小写选项时，此选项才适用。

- 保留源数据库中的参数名称。

要在转换后的代码中为参数名称添加双引号，请选择保留原始参数名称。

将 SQL Server 分区转换为 PostgreSQL 版本 10 分区

将 Microsoft SQL Server 数据库转换为 Amazon Aurora PostgreSQL 兼容版本 (Aurora PostgreSQL) 或适用于 PostgreSQL 的 Amazon Relational Database Service (Amazon RDS for PostgreSQL) 时，请注意以下几点：

在 SQL Server 中，使用分区函数创建分区。从 SQL Server 分区表转换为 PostgreSQL 版本 10 分区表时，请注意一些潜在问题：

- SQL Server 允许使用无 NOT NULL 约束的列为表分区。在此情况下，所有 NULL 值都将转到最左边的分区。PostgreSQL 不支持 RANGE 分区采用 NULL 值。
- SQL Server 允许为分区表创建主键和唯一键。对于 PostgreSQL，直接为每个分区创建主键或唯一键。因此，迁移到 PostgreSQL 时，必须从父表中删除 PRIMARY 或 UNIQUE KEY 约束。生成的键名称采用格式 `<original_key_name>_<partition_number>`。
- SQL Server 允许创建进出分区表的外键约束。PostgreSQL 不支持引用分区表的外键。此外，PostgreSQL 不支持从一个分区表到另一个表的外键引用。
- SQL Server 允许为分区表创建索引。对于 PostgreSQL，应直接为每个分区创建一个索引。因此，迁移到 PostgreSQL 时，索引必须从其父表中删除。生成的索引名称采用格式 `<original_index_name>_<partition_number>`。
- PostgreSQL 不支持分区索引。

迁移注意事项

将 SQL Server 架构迁移到 PostgreSQL 时要考虑的一些事项：

- 在 PostgreSQL 中，架构中所有对象 (包括索引) 的名称必须是唯一的。索引名称在基表的架构中必须是唯一的。在 SQL Server 中，索引名称可与其他表的相同。

  为确保索引名称的唯一性，AWS SCT 在索引名称不唯一时提供了生成唯一索引名称的选项。为此，请选择项目属性中的 Generate unique index names (生成唯一索引名称)。默认情况下，此选项处于启用状态。如果此选项处于启用状态，将使用格式 IX_table_name_index_name 创建唯一索引名称。如果此选项处于禁用状态，则不会更改索引名称。

- 一个 GOTO 语句和一个标签可用于更改语句的运行顺序。将跳过接在 GOTO 语句后的任何 Transact-SQL 语句并且处理将在标签处继续。GOTO 语句和标签可在过程、批处理或语句块中的任意位置使用。GOTO 语句也可以嵌套。

  PostgreSQL 不使用 GOTO 语句。当 AWS SCT 转换包含 GOTO 语句的代码时，它将转换此语句以使用 BEGIN…END 或 LOOP…END LOOP 语句。在下表中可以找到有关 AWS SCT 如何转换 GOTO 语句的示例。

  SQL Server GOTO 语句和已转换的 PostgreSQL 语句

SQL Server 语句	PostgreSQL 语句

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

SQL Server 语句	PostgreSQL 语句
BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	BEGIN .... statement1; .... label1: LOOP statement2; .... CONTINUE label1; EXIT label1; END LOOP; statement3; .... statement4; .... END

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

- PostgreSQL 不支持 MERGE 语句。AWS SCT 通过以下方式模拟 MERGE 语句的行为：
  - 通过 INSERT ON CONFLICT 结构。
  - 通过使用 UPDATE FROM DML 语句，例如没有 WHEN NOT MATCHED 子句的 MERGE。
  - 通过使用 CURSOR（如带有 DELETE 子句的 MERGE）或复杂的 MERGE ON 条件语句。
- 当 Amazon RDS 为目标时，AWS SCT 可将数据库触发器添加到对象树中。

- 当 Amazon RDS 为目标时，AWS SCT 可将服务器级别触发器添加到对象树中。
- SQL Server 会自动创建和管理 `deleted` 和 `inserted` 表。您可以使用这些临时的、内存驻留表测试某些数据修改的效果并为 DML 触发器操作设置条件。AWS SCT 可以在 DML 触发器语句中转换这些表的用法。
- 当 Amazon RDS 为目标时，AWS SCT 可将链接服务器添加到对象树中。
- 在从 Microsoft SQL Server 迁移到 PostgreSQL 时，内置的 SUSER_SNAME 函数进行如下转换：
  - SUSER_SNAME – 返回与安全标识号 (SID) 关联的登录名。
  - SUSER_SNAME(<server_user_sid>) – 不受支持。
  - SUSER_SNAME() CURRENT_USER – 返回当前执行上下文的用户名。
  - SUSER_SNAME(NULL) – 返回 NULL。
- 支持转换表值函数。表值函数返回一个表，并且可在查询中代替表。
- PATINDEX 在所有有效的文本和字符数据类型上返回指定表达式中模式的第一个匹配项的起始位置。如果找不到该模式，则返回零。当从 SQL Server 转换到 Amazon RDS for PostgreSQL 时，AWS SCT 会将使用 PATINDEX 的应用程序代码替换为 aws_sqlserver_ext.patindex (<模式字符>, <表达式字符变体>) 。
- 在 SQL Server 中，用户定义的表类型是表示表结构的定义的类型。可以使用用户定义的表类型来声明存储过程或函数的表值参数。还可以使用用户定义的表类型声明要在批处理中或者存储过程或函数的主体中使用的表变量。AWS SCT 通过创建临时表在 PostgreSQL 中模拟了此类型。

从 SQL Server 转换到 PostgreSQL 时，AWS SCT 会将 SQL Server 系统对象转换为 PostgreSQL 中的可识别对象。下表显示了如何转换系统对象。

MS SQL Server 使用案例	PostgreSQL 替代项
SYS.SCHEMAS	AWS_SQLSERVER_EXT.SYS_SCHEMAS
SYS.TABLES	AWS_SQLSERVER_EXT.SYS_TABLES
SYS.VIEWS	AWS_SQLSERVER_EXT.SYS_VIEWS
SYS.ALL_VIEWS	AWS_SQLSERVER_EXT.SYS_ALL_VIEWS
SYS.TYPES	AWS_SQLSERVER_EXT.SYS_TYPES
SYS.COLUMNS	AWS_SQLSERVER_EXT.SYS_COLUMNS

MS SQL Server 使用案例	PostgreSQL 替代项
SYS.ALL_COLUMNS	AWS_SQLSERVER_EXT.SYS_ALL_COLUMNS
SYS.FOREIGN_KEYS	AWS_SQLSERVER_EXT.SYS_FOREIGN_KEYS
SYS.SYSFOREIGNKEYS	AWS_SQLSERVER_EXT.SYS_SYSFOREIGNKEYS
SYS.FOREIGN_KEY_COLUMNS	AWS_SQLSERVER_EXT.SYS_FOREIGN_KEY_COLUMNS
SYS.KEY_CONSTRAINTS	AWS_SQLSERVER_EXT.SYS_KEY_CONSTRAINTS
SYS.IDENTITY_COLUMNS	AWS_SQLSERVER_EXT.SYS_IDENTITY_COLUMNS
SYS.PROCEDURES	AWS_SQLSERVER_EXT.SYS_PROCEDURES
SYS.INDEXES	AWS_SQLSERVER_EXT.SYS_INDEXES
SYS.SYSINDEXES	AWS_SQLSERVER_EXT.SYS_SYSINDEXES
SYS.OBJECTS	AWS_SQLSERVER_EXT.SYS_OBJECTS
SYS.ALL_OBJECTS	AWS_SQLSERVER_EXT.SYS_ALL_OBJECTS
SYS.SYSOBJECTS	AWS_SQLSERVER_EXT.SYS_SYSOBJECTS
SYS.SQL_MODULES	AWS_SQLSERVER_EXT.SYS_SQL_MODULES
SYS.DATABASES	AWS_SQLSERVER_EXT.SYS_DATABASES
INFORMATION_SCHEMA.SCHEMATA	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_SCHEMATA
INFORMATION_SCHEMA.VIEWS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_VIEWS
INFORMATION_SCHEMA.TABLES	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLES

MS SQL Server 使用案例	PostgreSQL 替代项
INFORMATION_SCHEMA.COLUMNS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_COLUMNS
INFORMATION_SCHEMA.CHECK_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CHECK_CONSTRAINTS
INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_REFERENTIAL_CONSTRAINTS
INFORMATION_SCHEMA.TABLE_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLE_CONSTRAINTS
INFORMATION_SCHEMA.KEY_COLUMN_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_KEY_COLUMN_USAGE
INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_TABLE_USAGE
INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_COLUMN_USAGE
INFORMATION_SCHEMA.ROUTINES	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_ROUTINES
SYS.SYSPROCESSES	AWS_SQLSERVER_EXT.SYS_SYSPROCESSES
sys.system_objects	AWS_SQLSERVER_EXT.SYS_SYSTEM_OBJECTS

使用 AWS SCT 扩展包在 PostgreSQL 中模拟 SQL Server Agent

SQL Server Agent 是运行 SQL Server 作业的 Microsoft Windows 服务。SQL Server Agent 可以根据计划、为响应特定事件或者按需运行作业。有关 SQL Server Agent 的详细信息，请参阅 Microsoft 技术文档。

PostgreSQL 没有 SQL Server Agent 的等效服务。要模拟 SQL Server Agent 功能，AWS SCT 会创建一个扩展包。此扩展包使用 AWS Lambda 和 Amazon CloudWatch。AWS Lambda 实现用于管理计划和运行作业的接口。Amazon CloudWatch 维护计划规则。

AWS Lambda 和 Amazon CloudWatch 使用 JSON 参数进行交互。此 JSON 参数具有以下结构。

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

在前面的示例中，*mode* 是任务的类型，*list of parameters* 是一组取决于任务类型的参数。此外，*procedure name* 是任务完成后运行的过程的名称。

AWS SCT 使用一个 Lambda 函数控制和运行作业。CloudWatch 规则开始运行作业，并提供启动任务所需的必要信息。当 CloudWatch 规则触发时，它会使用规则中的参数启动 Lambda 函数。

要创建调用过程的简单作业，请使用以下格式。

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

要创建多个步骤作业，请使用以下格式。

```
{
    "mode": "run_job",
    "parameters": {
        "job_name": "Job1",
        "enabled": "true",
        "start_step_id": 1,
        "notify_level_email": [0|1|2|3],
        "notify_email": email,
        "delete_level": [0|1|2|3],
```

```
        "job_callback": "ProcCallBackJob(job_name, code, message)",
        "step_callback": "ProcCallBackStep(job_name, step_id, code, message)"
    },
    "steps": [
        {
            "id":1,
            "cmd": "ProcStep1",
            "cmdexec_success_code": 0,
            "on_success_action": [|2|3|4],
            "on_success_step_id": 1,
            "on_fail_action": 0,
            "on_fail_step_id": 0,
            "retry_attempts": number,
            "retry_interval": number
        },
        {
            "id":2,
            "cmd": "ProcStep2",
            "cmdexec_success_code": 0,
            "on_success_action": [1|2|3|4],
            "on_success_step_id": 0,
            "on_fail_action": 0,
            "on_fail_step_id": 0,
            "retry_attempts": number,
            "retry_interval": number
        },
        ...
]
}
```

为了在 PostgreSQL 中模拟 SQL Server Agent 行为，AWS SCT 扩展包还会创建了以下表格和过程。

在 PostgreSQL 中模拟 SQL Server Agent 的表

为了模拟 SQL Server Agent，扩展包使用以下表：

sysjobs

存储有关作业的信息。

sysjobsteps

存储有关作业步骤的信息。

sysschedules

存储有关作业计划的信息。

sysjobschedules

存储各个作业的计划信息。

sysjobhistory

存储有关计划作业运行的信息。

在 PostgreSQL 中模拟 SQL Server Agent 的过程

要模拟 SQL Server Agent，扩展包使用以下过程：

sp_add_job

添加新作业。

sp_add_jobstep

向作业添加步骤。

sp_add_schedule

在 Amazon CloudWatch 中创建新的计划规则。您可以将此计划用于任意数量的作业。

sp_attach_schedule

设置所选作业的计划。

sp_add_jobschedule

在 Amazon CloudWatch 中为作业创建计划规则并为该规则设定目标。

sp_update_job

更新先前创建的作业的属性。

sp_update_jobstep

更新作业中步骤的属性。

sp_update_schedule

更新 Amazon CloudWatch 中计划规则的属性。

sp_update_jobschedule

更新指定作业计划的属性。

sp_delete_job

删除作业。

sp_delete_jobstep

从作业中删除一个作业步骤。

sp_delete_schedule

删除计划。

sp_delete_jobschedule

从 Amazon CloudWatch 中删除指定作业的计划规则。

sp_detach_schedule

移除计划和作业之间的关联。

get_jobs、update_job

与 AWS Elastic Beanstalk 交互的内部过程。

sp_verify_job_date、sp_verify_job_time、sp_verify_job、sp_verify_jobstep、sp_verify_schedule、sp_verify_j

检查设置的内部过程。

在 PostgreSQL 中模拟 SQL Server Agent 的过程的语法

扩展包中的 `aws_sqlserver_ext.sp_add_job` 过程模拟 `msdb.dbo.sp_add_job` 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

```
par_delete_level integer = 0,
inout par_job_id integer = NULL::integer,
par_originating_server varchar = NULL::character varying,
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sp_add_jobstep 过程模拟 msdb.dbo.sp_add_jobstep 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sp_add_schedule 过程模拟 msdb.dbo.sp_add_schedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

```
par_schedule_name varchar,
par_enabled smallint = 1,
par_freq_type integer = 0,
par_freq_interval integer = 0,
par_freq_subday_type integer = 0,
par_freq_subday_interval integer = 0,
```

```
par_freq_relative_interval integer = 0,
par_freq_recurrence_factor integer = 0,
par_active_start_date integer = NULL::integer,
par_active_end_date integer = 99991231,
par_active_start_time integer = 0,
par_active_end_time integer = 235959,
par_owner_login_name varchar = NULL::character varying,
*inout par_schedule_uid char = NULL::bpchar,*
inout par_schedule_id integer = NULL::integer,
par_originating_server varchar = NULL::character varying,
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sp_attach_schedule 过程模拟 msdb.dbo.sp_attach_schedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sp_add_jobschedule 过程模拟 msdb.dbo.sp_add_jobschedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

```
par_automatic_post smallint = 1,
inout par_schedule_uid char = NULL::bpchar,
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sp_delete_job 过程模拟 msdb.dbo.sp_delete_job 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sp_delete_jobstep 过程模拟 msdb.dbo.sp_delete_jobstep 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sp_delete_jobschedule 过程模拟 msdb.dbo.sp_delete_jobschedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sp_delete_schedule 过程模拟 msdb.dbo.sp_delete_schedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

```
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
```

```
par_force_delete smallint = 0,
par_automatic_post smallint = 1,
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sp_detach_schedule 过程模拟 msdb.dbo.sp_detach_schedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sp_update_job 过程模拟 msdb.dbo.sp_update_job 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sp_update_jobschedule 过程模拟 msdb.dbo.sp_update_jobschedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

扩展包中的 `aws_sqlserver_ext.sp_update_jobstep` 过程模拟 `msdb.dbo.sp_update_jobstep` 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

```
par_proxy_id integer = NULL::integer
par_proxy_name varchar = NULL::character varying
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sp_update_schedule 过程模拟 msdb.dbo.sp_update_schedule 过程。有关源 SQL Server Agent 过程的详细信息，请参阅 Microsoft 技术文档。

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

使用在 PostgreSQL 中模拟 SQL Server Agent 的过程的示例

要添加新作业，请使用如下所示的 aws_sqlserver_ext.sp_add_job 过程。

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

要添加新的作业步骤，请使用如下所示的 aws_sqlserver_ext.sp_add_jobstep 过程。

```
SELECT * FROM aws_sqlserver_ext.sp_add_jobstep (
    par_job_name := 'test_job',
    par_step_id := 1::smallint,
```

```
    par_step_name := 'test_job_step1',
    par_subsystem := 'TSQL',
    par_command := 'EXECUTE [dbo].[PROC_TEST_JOB_STEP1];',
    par_server := NULL,
    par_database_name := 'GOLD_TEST_SS');
```

要添加简单的计划，请使用如下所示的 aws_sqlserver_ext.sp_add_schedule 过程。

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

要为作业设置计划，请使用如下所示的 aws_sqlserver_ext.sp_attach_schedule 过程。

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

要为作业创建计划，请使用如下所示的 aws_sqlserver_ext.sp_add_jobschedule 过程。

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

在 PostgreSQL 中模拟 SQL Server Agent 的使用案例示例

如果源数据库代码使用 SQL Server Agent 运行作业，则可以使用 AWS SCT 的 SQL Server 到 PostgreSQL 扩展包将此代码转换为 PostgreSQL。扩展包使用 AWS Lambda 函数模拟 SQL Server Agent 的行为。

您可以创建新的 AWS Lambda 函数或注册现有函数。

创建新 AWS Lambda 函数

1. 在 AWS SCT 的目标数据库树中，打开上下文（右键单击）菜单，然后选择应用扩展包，然后选择 PostgreSQL。

   扩展包向导随即出现。

2. 在 SQL Server Agent 模拟服务选项卡上，执行以下操作：

   - 选择创建 AWS Lambda 函数。
   - 在数据库登录名中，输入目标数据库用户名。
   - 在数据库密码中，输入您在上一步中输入的用户名的密码。
   - 对于 Python 库文件夹，请输入 Python 库文件夹的路径。
   - 选择创建 AWS Lambda 函数，然后选择下一步。

注册之前部署的 AWS Lambda 函数

- 在目标数据库上运行以下脚本。

```
SELECT
    FROM aws_sqlserver_ext.set_service_setting(
        p_service := 'JOB',
        p_setting := 'LAMBDA_ARN',
        p_value := ARN)
```

在上述示例中，*ARN* 是已部署 AWS Lambda 函数的 Amazon 资源名称（ARN）。

以下示例创建了一个由一个步骤组成的简单任务。此任务每五分钟运行一次先前创建的 `job_example` 函数。此函数将记录插入 `job_example_table` 表中。

创建这个简单的任务

1. 使用 `aws_sqlserver_ext.sp_add_job` 函数创建作业，如下所示。

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

2. 使用 aws_sqlserver_ext.sp_add_jobstep 函数创建任务步骤，如下所示。

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

作业步骤指定函数的用途。

3. 使用 aws_sqlserver_ext.sp_add_jobschedule 函数为作业创建计划程序，如下所示。

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

作业步骤指定函数的用途。

要删除此作业，请使用如下所示的 aws_sqlserver_ext.sp_delete_job 函数。

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

使用 AWS SCT 扩展包在 PostgreSQL 中模拟 SQL Server 数据库邮件

您可以使用 SQL Server 数据库邮件将从 SQL Server 数据库引擎或 Azure SQL 托管实例向用户发送电子邮件。这些电子邮件消息可以包含查询结果，也可以包含来自网络上任何资源的文件。有关 SQL Server 数据库邮件的更多信息，请参阅 Microsoft 技术文档。

PostgreSQL 没有与 SQL Server 数据库邮件等效的内容。要模拟 SQL Server 数据库邮件功能，AWS SCT 会创建一个扩展包。此扩展包使用 AWS Lambda 和 Amazon Simple Email Service (Amazon SES)。AWS Lambda 为用户提供了一个与 Amazon SES 电子邮件发送服务进行交互的接口。要设置此交互，请添加 Lambda 函数的 Amazon 资源名称（ARN）。

对于新的电子邮件账户，请使用以下命令。

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

要将 Lambda 函数的 ARN 添加到现有电子邮件账户，请使用以下命令。

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

在上述示例中，*ARN* 是 Lambda 函数的 ARN。

为了在 PostgreSQL 中模拟 SQL Server 数据库邮件行为，AWS SCT 扩展包使用了以下表、视图和过程。

在 PostgreSQL 中模拟 SQL Server 数据库邮件的表

为了模拟 SQL Server 数据库邮件，扩展包使用以下表：

sysmail_account

存储有关电子邮件账户的信息。

sysmail_profile

存储有关用户配置文件的信息。

sysmail_server

存储有关电子邮件服务器的信息。

sysmail_mailitems

存储电子邮件消息列表。

sysmail_attachments

每个电子邮件附件包含一行。

sysmail_log

存储有关发送电子邮件消息的服务信息。

sysmail_profileaccount

存储有关用户配置文件和电子邮件账户的信息。

在 PostgreSQL 中模拟 SQL Server 数据库邮件的视图

要模拟 SQL Server 数据库邮件，AWS SCT 在 PostgreSQL 数据库中创建以下视图以确保兼容性。扩展包不使用这些视图，但转换后的代码可以查询它们。

sysmail_allitems

包括所有电子邮件的列表。

sysmail_faileditems

包括无法发送的电子邮件列表。

sysmail_sentitems

包括已发送电子邮件的列表。

sysmail_unsentitems

包括尚未发送的电子邮件列表。

sysmail_mailattachments

包括附件列表。

在 PostgreSQL 中模拟 SQL Server 数据库邮件的过程

要模拟 SQL Server 数据库邮件，扩展包使用以下过程：

sp_send_dbmail

向指定的收件人发送电子邮件。

sysmail_add_profile_sp

创建新的用户配置文件

sysmail_add_account_sp

创建用于存储简单邮件传输协议 (SMTP) 凭证等信息的新电子邮件帐户。

sysmail_add_profileaccount_sp

将电子邮件帐户添加到指定的用户配置文件中。

sysmail_update_profile_sp

更改用户配置文件的属性，例如描述、名称等。

sysmail_update_account_sp

更改现有电子邮件帐户中的信息。

sysmail_update_profileaccount_sp

更新指定用户配置文件中的电子邮件帐户信息。

sysmail_delete_profileaccount_sp

从指定的用户配置文件中删除电子邮件帐户。

sysmail_delete_account_sp

删除电子邮件账户。

sysmail_delete_profile_sp

删除用户配置文件

sysmail_delete_mailitems_sp

从内部表格中删除电子邮件。

sysmail_help_profile_sp

显示有关用户配置文件的信息。

sysmail_help_account_sp

显示有关电子邮件帐户的信息。

sysmail_help_profileaccount_sp

显示有关与用户配置文件关联的电子邮件账户的信息。

sysmail_dbmail_json

为 AWS Lambda 函数生成 JSON 请求的内部过程。

sysmail_verify_profile_sp、sysmail_verify_account_sp、sysmail_verify_addressparams_sp

检查设置的内部过程。

sp_get_dbmail、sp_set_dbmail、sysmail_dbmail_xml

已弃用的内部过程。

在 PostgreSQL 中模拟 SQL Server 数据库邮件的过程的语法

扩展包中的 `aws_sqlserver_ext.sp_send_dbmail` 过程模拟 `msdb.dbo.sp_send_dbmail` 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

```
par_query_no_truncate smallint = 0,
par_query_result_no_padding smallint = 0,
out par_mailitem_id integer,
par_from_address text = NULL::text,
par_reply_to text = NULL::text,
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sysmail_delete_mailitems_sp 过程模拟 msdb.dbo.sysmail_delete_mailitems_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_add_profile_sp 过程模拟 msdb.dbo.sysmail_add_profile_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_add_account_sp 过程模拟 msdb.dbo.sysmail_add_account_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

```
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sysmail_add_profileaccount_sp 过程模拟 msdb.dbo.sysmail_add_profileaccount_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_help_profile_sp 过程模拟 msdb.dbo.sysmail_help_profile_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_update_profile_sp 过程模拟 msdb.dbo.sysmail_update_profile_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_delete_profile_sp 过程模拟 msdb.dbo.sysmail_delete_profile_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_force_delete smallint = 1,
out returncode integer
```

扩展包中的 aws_sqlserver_ext.sysmail_help_account_sp 过程模拟 msdb.dbo.sysmail_help_account_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_update_account_sp 过程模拟 msdb.dbo.sysmail_update_account_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_delete_account_sp 过程模拟 msdb.dbo.sysmail_delete_account_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_help_profileaccount_sp 过程模拟 msdb.dbo.sysmail_help_profileaccount_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_update_profileaccount_sp 过程模拟 msdb.dbo.sysmail_update_profileaccount_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

扩展包中的 aws_sqlserver_ext.sysmail_delete_profileaccount_sp 过程模拟 msdb.dbo.sysmail_delete_profileaccount_sp 过程。有关源 SQL Server 数据库邮件过程的更多信息，请参阅 Microsoft 技术文档。

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

使用在 PostgreSQL 中模拟 SQL Server 数据库邮件的过程的示例

要发送电子邮件，请使用如下所示的 aws_sqlserver_ext.sp_send_dbmail 过程。

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

以下示例演示如何通过查询结果发送电子邮件。

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

以下代码示例展示如何通过 HTML 代码发送电子邮件。

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

要删除电子邮件，请使用如下所示的 aws_sqlserver_ext.sysmail_delete_mailitems_sp 过程。

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

下面的示例说明如何删除最旧的电子邮件。

```
PERFORM sysmail_delete_mailitems_sp (
    par_sent_before := '31.12.2015'
);
```

以下示例将说明如何删除所有无法发送的电子邮件。

```
PERFORM sysmail_delete_mailitems_sp (
    par_sent_status := 'failed'
);
```

要创建新的用户配置文件，请使用如下所示的 aws_sqlserver_ext.sysmail_add_profile_sp 过程。

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

以下示例说明如何创建新的配置文件并将唯一配置文件标识符保存在变量中。

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

SELECT var_profileId;
```

要创建新的电子邮件帐户，请使用如下所示的 aws_sqlserver_ext.sysmail_add_account_sp 过程。

```
PERFORM sysmail_add_account_sp (
    par_account_name :='Audit Account',
    par_email_address := 'dba@rusgl.info',
    par_display_name := 'Test Automated Mailer',
    par_description := 'Account for administrative e-mail.',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'arn:aws:lambda:us-west-2:555555555555:function:pg_v3'
);
```

要将电子邮件帐户添加到用户配置文件中，请按以下 aws_sqlserver_ext.sysmail_add_profileaccount_sp 过程操作。

```
PERFORM sysmail_add_profileaccount_sp (
```

```
    par_account_name := 'Administrator',
    par_account_name := 'Audit Account',
    par_sequence_number := 1
);
```

在 PostgreSQL 中模拟 SQL Server 数据库邮件的使用案例示例

如果源数据库代码使用 SQL Server 数据库邮件发送电子邮件，则可以使用 AWS SCT 扩展包将此代码转换为 PostgreSQL。

从 PostgreSQL 数据库发送一封电子邮件

1. 创建和配置 AWS Lambda 函数。
2. 应用 AWS SCT 扩展包
3. 使用如下所示的 `sysmail_add_profile_sp` 函数创建用户配置文件。
4. 使用如下所示的 `sysmail_add_account_sp` 函数创建电子邮件帐户。
5. 使用如下所示的 `sysmail_add_profileaccount_sp` 函数，将此电子邮件帐户添加到用户配置文件中。

```
CREATE OR REPLACE FUNCTION aws_sqlserver_ext.
proc_dbmail_settings_msdb()
RETURNS void
AS
$BODY$
BEGIN
PERFORM aws_sqlserver_ext.sysmail_add_profile_sp(
    par_profile_name := 'Administrator',
    par_description := 'administrative mail'
);
PERFORM aws_sqlserver_ext.sysmail_add_account_sp(
    par_account_name := 'Audit Account',
    par_description := 'Account for administrative e-mail.',
    par_email_address := 'dba@rusgl.info',
    par_display_name := 'Test Automated Mailer',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'your_ARN'
);
PERFORM aws_sqlserver_ext.sysmail_add_profileaccount_sp(
    par_profile_name := 'Administrator',
    par_account_name := 'Audit Account',
    par_sequence_number := 1
```

```
);
END;
$BODY$
LANGUAGE plpgsql;
```

6. 使用如下所示的 sp_send_dbmail 函数发送电子邮件。

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

要查看有关所有用户配置文件的信息，请按以下 sysmail_help_profile_sp 过程操作。

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

以下示例显示有关特定用户配置文件的信息。

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

要查看有关所有电子邮件帐户的信息，请使用如下所示的 sysmail_help_account_sp 过程。

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

以下示例显示有关特定电子邮件账户的信息。

```
select from aws_sqlserver_ext.sysmail_help_account_sp(par_account_id := 1);
```

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

要查看与用户配置文件关联的所有电子邮件帐户的信息，请按以下 sysmail_help_profileaccount_sp 过程操作。

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

以下示例按标识符、配置文件名称或账户名筛选记录。

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

要更改用户配置文件名称或描述，请按以下 sysmail_update_profile_sp 过程操作。

```
select aws_sqlserver_ext.sysmail_update_profile_sp(
    par_profile_id := 2,
    par_profile_name := 'New profile name'
);
```

要更改电子邮件帐户设置，请使用如下所示的 ysmail_update_account_sp 过程。

```
select from aws_sqlserver_ext.sysmail_update_account_sp (
    par_account_name := 'Audit Account',
    par_mailserver_name := 'arn:aws:lambda:region:XXXXXXXXXXXX:function:func_test',
    par_mailserver_type := 'AWSLAMBDA'
);
```

将 SQL Server 转换为 Amazon RDS for SQL Server

将 SQL Server 架构和代码迁移到 Amazon RDS for SQL Server 时要考虑的一些事项：

- AWS SCT 可以转换 SQL Server Agent 以在 Amazon RDS for SQL Server 数据库实例上提供计划、提醒和作业。转换后，可以将 Amazon RDS for SQL Server 数据库实例与 SQL Server Reporting Services (SSRS)、SQL Server Analysis Services (SSAS) 和 SQL Server Integration Services (SSIS) 结合使用。

- Amazon RDS 当前不支持 SQL Server Service Broker 或其他需要您运行 CREATE ENDPOINT 命令的 T-SQL 终端节点。
- Amazon RDS 对链接的服务器具有有限的支持。在转换使用链接服务器的 SQL Server 应用程序代码时，AWS SCT 将转换应用程序代码。但是，请确保先查看使用链接服务器的对象的行为，然后再运行转换的代码。
- 使用“始终打开”。
- AWS SCT 评估报告提供了转换的服务器指标。这些有关 SQL Server 实例的指标包括：
  - 使用了数据镜像。
  - 配置了 SQL Server 日志传输。
  - 使用了故障转移群集。
  - 配置了数据库邮件。
  - 使用了全文搜索服务。Amazon RDS for SQL Server 具有有限的全文搜索，并且不支持语义搜索。
  - 安装了 Data Quality Service (DQS)。Amazon RDS 不支持 DQS，因此我们建议您在 Amazon EC2 实例上安装 SQL Server。

将 RDS for SQL Server 作为目标的权限

要迁移到 RDS for SQL Server，请创建一个数据库用户，然后为每个数据库授予所需的权限。您可以使用以下代码示例：

```
CREATE LOGIN user_name WITH PASSWORD 'your_password';

USE db_name
CREATE USER user_name FOR LOGIN user_name
GRANT VIEW DEFINITION TO user_name
GRANT VIEW DATABASE STATE TO user_name
GRANT CREATE SCHEMA TO user_name;
GRANT CREATE TABLE TO user_name;
GRANT CREATE VIEW TO user_name;
GRANT CREATE TYPE TO user_name;
GRANT CREATE DEFAULT TO user_name;
GRANT CREATE FUNCTION TO user_name;
GRANT CREATE PROCEDURE TO user_name;
GRANT CREATE ASSEMBLY TO user_name;
GRANT CREATE AGGREGATE TO user_name;
GRANT CREATE FULLTEXT CATALOG TO user_name;
GRANT CREATE SYNONYM TO user_name;
```

```
GRANT CREATE XML SCHEMA COLLECTION TO user_name;
```

在前面的示例中，将 *user_name* 替换为用户名。然后，将 *db_name* 替换为目标数据库名称。最后，将 *your_password* 替换为安全密码。

AWS Schema Conversion Tool 的数据仓库源

AWS SCT 可以将以下源数据仓库的架构转换为支持的目标。有关权限、连接以及 AWS SCT 可转换的用于目标数据库或数据仓库的内容的信息，请参阅以下详细信息。

主题

将 Amazon Redshift 用作 AWS SCT 的源

您可以使用 AWS SCT 优化 Amazon Redshift 集群。AWS SCT 为您提供有关为 Amazon Redshift 集群选择分配键和排序键的建议。您可以将 Amazon Redshift 优化项目视为一个源和目标指向不同的 Amazon Redshift 集群的 AWS SCT 项目。

将 Amazon Redshift 用作源数据库的权限

下面列出了将 Amazon Redshift 用作源所需的权限：

- USAGE ON SCHEMA *<schema_name>*
- SELECT ON ALL TABLES IN SCHEMA *<schema_name>*
- SELECT ON PG_CATALOG.PG_STATISTIC

- SELECT ON SVV_TABLE_INFO
- SELECT ON TABLE STV_BLOCKLIST
- SELECT ON TABLE STV_TBL_PERM
- SELECT ON SYS_SERVERLESS_USAGE
- SELECT ON PG_DATABASE_INFO
- SELECT ON PG_STATISTIC

在前面的示例中，将 *<schema_name>* 占位符替换为源架构的名称。

有关将 Amazon Redshift 作为目标所需的权限，请参阅 将 Amazon Redshift 作为目标的权限。

连接到作为源的 Amazon Redshift

使用 AWS Schema Conversion Tool 按照以下过程连接到 Amazon Redshift 源数据库。

连接到 Amazon Redshift 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Amazon Redshift，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要输入 Amazon Redshift 源数据库的连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。

参数	操作
数据库。	输入 Amazon Redshift 数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层（SSL）连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 验证服务器证书：选择此选项以使用信任存储验证服务器证书。 • 信任存储：包含证书的信任存储的位置。要使此位置出现在此处，请务必将其添加到全局设置中。 有关对 Amazon Redshift 的 SSL 支持的更多信息，请参阅配置连接的安全选项。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
Redshift 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

Amazon Redshift 优化设置

要编辑 Amazon Redshift 优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Amazon Redshift，然后选择 Amazon Redshift – Amazon Redshift。AWS SCT 显示 Amazon Redshift 优化的所有可用设置。

AWS SCT 中的 Amazon Redshift 优化设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过您的 Amazon Redshift 集群可以存储的表数量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 选择迁移策略。

  AWS 建议使用不同的集群作为优化项目的源和目标。在 Amazon Redshift 优化过程开始之前，需要创建源 Amazon Redshift 集群的副本。您可以将源数据包含到此副本中，也可以创建一个空集群。

  对于迁移策略，请选择迁移到副本以将源集群中的数据包含在目标集群中。

  对于迁移策略，请选择迁移到干净的画面以查看优化建议。接受这些建议后，请将源数据迁移到目标集群。

- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择了使用压缩编码选项时，才能选择此选项。

- 使用自动表优化。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅用于自动表格优化，请在左侧窗格中选择优化策略。然后选择使用 Amazon Redshift 自动调整表格，并在初始键选择策略中选择无。

- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：

  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

  您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除偏斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

使用 Azure Synapse Analytics 作为 AWS SCT 的源

你可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Azure Synapse Analytics 转换为 Amazon Redshift。

作为源数据库的 Azure Synapse Analytics 的权限

使用 Azure Synapse Analytics 数据仓库作为源需要以下权限：

- VIEW DEFINITION
- VIEW DATABASE STATE

对要转换架构的每个数据库应用这种权限。

连接到作为源的 Azure Synapse Analytics

通过以下过程使用 AWS Schema Conversion Tool 连接到 Azure Synapse Analytics 数据仓库。

连接到作为源的 Azure Synapse Analytics 数据仓库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Azure Synapse Analytics，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 Azure Synapse Analytics 数据仓库的连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名服务 (DNS) 名称或 IP 地址。

参数	操作
SQL 池	输入 Azure SQL 池的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层 (SSL) 连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 信任服务器证书：选择此选项以信任服务器证书。 • 信任存储：您在全局设置中设置的信任存储。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项可让您存储数据库密码并在不需要输入密码的情况下快速连接到数据库。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

Azure Synapse Analytics 到 Amazon Redshift 的转换设置

要编辑 Azure Synapse Analytics 到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Azure Synapse，然后选择 Azure Synapse – Amazon Redshift。AWS SCT 显示 Azure Synapse Analytics 到 Amazon Redshift 转换的所有可用设置。

AWS SCT 中的 Azure Synapse Analytics 到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 在 Amazon Redshift 中将源表的分区迁移到单独的表。为此，请选择使用 UNION ALL 视图，然后输入 AWS SCT 可以为单个源表创建的最大目标表数。

  Amazon Redshift 不支持表分区。要模拟此行为并加快查询运行速度，AWS SCT 可以将源表的每个分区迁移到 Amazon Redshift 中的单独表中。然后，AWS SCT 创建一个包含所有这些表中的数据的视图。

  AWS SCT 自动确定源表中的分区数量。根据源表分区的类型，此数量可能会超过您可以应用于 Amazon Redshift 集群的表配额。为避免达到此配额，请输入 AWS SCT 可以为单个源表的分区创建的最大目标表数。默认选项为 368 个表，它表示一年中 366 天的分区和以及 `NO RANGE` 和 `UNKNOWN` 分区的两个表。

- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

Azure Synapse Analytics 到 Amazon Redshift 转换优化设置

要编辑 Azure Synapse Analytics 到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Azure Synapse，然后选择 Azure Synapse – Amazon

Redshift。在左窗格中，选择优化策略。AWS SCT 显示 Azure Synapse Analytics 到 Amazon Redshift 的转换优化设置。

AWS SCT 中的 Azure Synapse Analytics 到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：
  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

  您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。
- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。
  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

  此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

使用 BigQuery 作为 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 BigQuery 转换到 Amazon Redshift。

作为源的 BigQuery 的权限

要使用 BigQuery 数据仓库作为 AWS SCT 的源，请创建一个服务账号。在 Google Cloud 中，应用程序使用服务账户进行授权的 API 调用。服务账户与用户账户不同。有关更多信息，请参阅《Google Cloud 身份和访问权限管理》文档中的服务账号。

请确保向您的服务账号授予以下角色：

- `BigQuery Admin`
- `Storage Admin`

`BigQuery Admin` 角色提供管理项目内所有资源的权限。AWS SCT 使用此角色在迁移项目中加载 BigQuery 元数据。

`Storage Admin` 角色授予对数据对象和存储桶的完全控制权。你可以在 `Cloud Storage` 下面找到这个角色。AWS SCT 使用此角色从 BigQuery 中提取数据，然后将其加载到 Amazon Redshift 中。

创建服务账号密钥文件

1. 登录 Google Cloud 管理控制台，网址为 https://console.cloud.google.com/。
2. 在 BigQuery API 页面上，选择启用。如果您看到 API 已启用，请跳过此步骤。
3. 在服务账号页面上，选择项目，然后选择创建服务账号。
4. 在服务账户详细信息页面上，在服务账户名称中输入描述性值。选择创建并继续。将打开授予此服务账号对项目的访问权限页面。
5. 在选择角色中，选择 BigQuery，然后选择 BigQuery 管理员。
6. 选择添加其他角色。在选择角色中，选择云存储，然后选择存储管理员。
7. 选择继续，然后选择完成。
8. 在服务账号页面上，选择您创建的服务账号。
9. 选择密钥，然后为添加密钥选择创建新密钥。
10. 选择 JSON，然后选择创建。选择用于保存私有密钥的文件夹，或者在浏览器中选择用于下载的默认文件夹。

要从 BigQuery 数据仓库中提取数据，AWS SCT 使用 Google Cloud Storage 存储桶文件夹。在开始数据迁移之前创建此存储桶。在创建本地任务对话框中输入 Google Cloud Storage 存储桶文件夹的路径。有关更多信息，请参阅创建、运行和监控 AWS SCT 任务。

连接到作为源的 BigQuery

使用 AWS Schema Conversion Tool 按照以下过程连接到 BigQuery 源项目。

连接到 BigQuery 源数据仓库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 BigQuery，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入 BigQuery 项目的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 在密钥路径中，输入服务账号密钥文件的路径。有关创建此文件的更多信息，请参阅 作为源的 BigQuery 的权限。
5. 选择测试连接以验证 AWS SCT 是否可以连接到源 BigQuery 项目。
6. 选择连接以连接到源 BigQuery 项目。

将 BigQuery 作为 AWS SCT 源的限制

将 BigQuery 作为 AWS SCT 的源时，以下限制适用：

- AWS SCT 不支持在分析函数中转换子查询。
- 您不能使用 AWS SCT 转换 BigQuery `SELECT AS STRUCT` 和 `SELECT AS VALUE` 语句。
- AWS SCT 不支持以下类型的函数的转换：
  - 近似聚合
  - 位
  - 调试
  - 联合查询
  - 地理位置
  - 哈希
  - 数学
  - 净值

  - 统计聚合
  - UUID
- AWS SCT 为字符串函数的转换提供了有限的支持。
- AWS SCT 不支持 UNNEST 运算符的转换。
- 无法在 AWS SCT 中转换相关的联接操作。
- AWS SCT 不支持 QUALIFY、WINDOW、LIMIT 和 OFFSET 子句的转换。
- 不能使用 AWS SCT 转换递归公用表表达式。
- AWS SCT 不支持转换带有 VALUES 子句内子查询的 INSERT 语句。
- AWS SCT 不支持嵌套字段和重复记录的 UPDATE 语句转换。
- 不能使用 AWS SCT 转换 STRUCT 和 ARRAY 数据类型。

BigQuery 到 Amazon Redshift 的转换设置

要编辑 BigQuery 到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中，选择 Google BigQuery，然后选择 Google BigQuery – Amazon Redshift。AWS SCT 显示 BigQuery 转换为 Amazon Redshift 的所有可用设置。

AWS SCT 中的 BigQuery 到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。
- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后

的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

BigQuery 到 Amazon Redshift 的转换优化设置

要编辑 BigQuery 到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中，选择 Google BigQuery，然后选择 Google BigQuery – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示将 BigQuery 转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中的 BigQuery 到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：

  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

  此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

将 Greenplum 数据库用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Greenplum 数据库转换到 Amazon Redshift。

将 Greenplum 数据库用作源的权限

下面列出了将 Greenplum 数据库用作源所需的权限：

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

在上述示例中，替换以下占位符：

- 将 *database_name* 替换为源数据库名称。
- 将 *schema_name* 替换为源架构的名称。
- 将 *table_name* 替换为源表的名称。

- 将 *sequence_name* 替换为序列名称的名称。

连接到作为源的 Greenplum 数据库

使用 AWS SCT 按照以下过程连接到 Greenplum 源数据库。

连接到 Greenplum 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 SAP ASE，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 Greenplum 源数据库凭证，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
数据库。	输入 Greenplum 数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。

参数	操作
使用 SSL	选择此选项以使用安全套接字层 (SSL) 连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 验证服务器证书：选择此选项以使用信任存储验证服务器证书。 • 信任存储：包含证书的信任存储的位置。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
Greenplum 数据库驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

Greenplum 到 Amazon Redshift 的转换设置

要编辑 Greenplum 到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Greenplum，然后选择 Greenplum – Amazon Redshift。AWS SCT 显示 Greenplum 转换到 Amazon Redshift 的所有可用设置。

AWS SCT 中的 Greenplum 到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 在 Amazon Redshift 中将源表的分区迁移到单独的表。为此，请选择使用 UNION ALL 视图，然后输入 AWS SCT 可以为单个源表创建的最大目标表数。

  Amazon Redshift 不支持表分区。要模拟此行为并加快查询运行速度，AWS SCT 可以将源表的每个分区迁移到 Amazon Redshift 中的单独表中。然后，AWS SCT 创建一个包含所有这些表中的数据的视图。

  AWS SCT 自动确定源表中的分区数量。根据源表分区的类型，此数量可能会超过您可以应用于 Amazon Redshift 集群的表配额。为避免达到此配额，请输入 AWS SCT 可以为单个源表的分区创建的最大目标表数。默认选项为 368 个表，它表示一年中 366 天的分区和以及 `NO RANGE` 和 `UNKNOWN` 分区的两个表。
- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

Greenplum 到 Amazon Redshift 的转换优化设置

要编辑 Greenplum 到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Greenplum，然后选择 Greenplum – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示 Greenplum 转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中 Greenplum 到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：

  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

  您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。
- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

  此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

将 Netezza 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Netezza 转换到 Amazon Redshift。

将 Netezza 用作源的权限

下面列出了将 Netezza 用作源所需的权限：

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

在上述示例中，替换以下占位符：

- 将 *database_name* 替换为源数据库名称。
- 将 *schema_name* 替换为源架构的名称。

AWS SCT 需要以下系统表和视图的访问权限。您可以授予对这些对象的访问权限，而不必授予对上述列表中的 `system.definition_schema.system view` 和 `system.definition_schema.system tables` 的访问权限。

- select on system.definition_schema._t_aggregate
- select on system.definition_schema._t_class
- select on system.definition_schema._t_constraint
- select on system.definition_schema._t_const_relattr
- select on system.definition_schema._t_database
- select on system.definition_schema._t_grpobj_priv

- select on system.definition_schema._t_grpusr
- select on system.definition_schema._t_hist_config
- select on system.definition_schema._t_object
- select on system.definition_schema._t_object_classes
- select on system.definition_schema._t_proc
- select on system.definition_schema._t_type
- select on system.definition_schema._t_user
- select on system.definition_schema._t_usrobj_priv
- select on system.definition_schema._vt_sequence
- select on system.definition_schema._v_aggregate
- select on system.definition_schema._v_constraint_depends
- select on system.definition_schema._v_database
- select on system.definition_schema._v_datatype
- select on system.definition_schema._v_dslice
- select on system.definition_schema._v_function
- select on system.definition_schema._v_group
- select on system.definition_schema._v_obj_relation
- select on system.definition_schema._v_obj_relation_xdb
- select on system.definition_schema._v_procedure
- select on system.definition_schema._v_relation_column
- select on system.definition_schema._v_relation_keydata
- select on system.definition_schema._v_relobjclasses
- select on system.definition_schema._v_schema_xdb
- select on system.definition_schema._v_sequence
- select on system.definition_schema._v_synonym
- select on system.definition_schema._v_system_info
- select on system.definition_schema._v_sys_constraint
- select on system.definition_schema._v_sys_object_dslice_info
- select on system.definition_schema._v_sys_user
- select on system.definition_schema._v_table

- select on system.definition_schema._v_table_constraint
- select on system.definition_schema._v_table_dist_map
- select on system.definition_schema._v_table_organize_column
- select on system.definition_schema._v_table_storage_stat
- select on system.definition_schema._v_user
- select on system.definition_schema._v_view
- select on system.information_schema._v_relation_column
- select on system.information_schema._v_table
- select on $hist_column_access_*

连接到作为源的 Netezza

使用 AWS Schema Conversion Tool 按照以下过程连接到 Netezza 源数据库。

连接到 Netezza 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Netezza，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 Netezza 源数据库的连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。

参数	操作
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
Netezza 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

配置持续的数据复制

转换 Netezza 数据库架构并将其应用于 Amazon Redshift 数据库后，您可以使用 AWS SCT 数据提取代理迁移数据。代理会提取数据并将其上载到 Amazon S3 存储桶。然后您可使用 AWS SCT 将数据从 Amazon S3 复制到 Amazon Redshift。

如果源数据库中的数据在迁移过程中发生了变化，则可以使用 AWS SCT 数据提取代理捕获正在进行的更改。然后，您可以在完成初始数据迁移后将这些正在进行的更改复制到目标数据库中。此过程称为持续的数据复制或更改数据捕获（CDC）。

为从 Netezza 迁移到 Amazon Redshift 配置持续的数据复制

1. 在源数据库中，创建一个历史数据库。您可在 Netezza 命令行界面（CLI）中使用以下代码示例。

```
nzhistcreatedb -d history_database_name -t query -v 1 -u load_user -o histdb_owner
 -p your_password
```

在前面的示例中，将 *history_database_name* 替换为历史数据库的名称。接下来，将 *load_user* 替换为您定义的用于将历史数据加载到数据库中的用户的名称。然后，将 *histdb_owner* 替换为您定义为历史数据库所有者的用户名。请确保您已经创建了此用户并授予了 CREATE DATABASE 权限。最后，将 *your_password* 替换为安全密码。

2. 配置历史日志记录。为此，请使用以下代码示例。

```
CREATE HISTORY CONFIGURATION history_configuration_name HISTTYPE QUERY
    DATABASE history_database_name USER load_user PASSWORD your_password COLLECT
 PLAN, COLUMN
    LOADINTERVAL 1 LOADMINTHRESHOLD 0 LOADMAXTHRESHOLD 0 STORAGELIMIT 25
    LOADRETRY 2 VERSION 1;
```

在前面的示例中，将 *history_configuration_name* 和 *history_database_name* 替换为历史配置的名称和历史数据库的名称。接下来，将 *load_user* 替换为您定义的用于将历史数据加载到数据库中的用户的名称。然后，将 *your_password* 替换为安全密码。

3. 授予历史数据库中所有表的读取权限。您可以使用以下代码示例授予 SELECT 权限。

```
GRANT SELECT ON history_database_name.ALL.TABLE TO your_user;
```

在前面的示例中，将 *history_database_name* 替换为历史数据库的名称。接下来，将 *your_user* 替换为具有使用 Netezza 数据库的最低权限的用户名。您可以在 AWS SCT 中使用该数据库用户的凭证。

4. 收集源架构中每个表的统计数据，以获取有关列基数的信息。您可使用以下命令在历史数据库中生成统计数据。

```
GENERATE STATISTICS on "schema_name"."table_name";
```

在前面的示例中，将 *schema_name* 和 *table_name* 替换为数据库架构名称和表名称。

5. 通过运行以下查询，确保您已完成先决条件：

```
SELECT COUNT(*)
 FROM history_database_name.history_schema_name."$hist_column_access_N";
```

在前面的示例中，将 *history_database_name* 和 *history_schema_name* 替换为历史数据库名称和架构名称。接下来，将 *N* 替换为历史数据库的版本号。有关历史数据库版本的更多信息，请参阅 IBM Netezza 文档。

6. 安装数据提取代理。有关更多信息，请参阅安装提取代理。

   确保 `settings.properties` 文件中所有提取器实例的 `{working.folder}` 参数都指向同一个文件夹。在这种情况下，提取器可以协调 CDC 会话并为所有子任务使用单个事务点。

7. 注册您的数据提取代理。有关更多信息，请参阅在中注册提取剂 AWS Schema Conversion Tool。

8. 创建 CDC 任务。有关更多信息，请参阅创建、运行和监控 AWS SCT 任务。

   a. 在 AWS SCT 中打开您的项目。在左窗格中，选择源表。打开上下文 (右键单击) 菜单，然后选择创建本地任务。

   b. 对于任务名称，请输入数据迁移任务的描述性名称。

   c. 对于迁移模式，选择提取、上传和复制。

   d. 选择启用 CDC。

   e. 选择 CDC 设置选项卡并定义 CDC 会话的范围和计划。

   f. 选择测试任务以验证是否可连接到您的工作文件夹、Amazon S3 存储桶和 Amazon Redshift 数据仓库。

   g. 选择创建以创建任务。

   h. 选择任务选项卡，从列表中选择您的任务，然后选择开始。

9. AWS SCT 任务在目标数据库上保持事务一致性。数据提取代理按事务 ID 顺序从源复制事务。

   如果您停止任何迁移会话或迁移会话失败，则 CDC 处理也会停止。

Netezza 到 Amazon Redshift 的转换设置

要编辑 Netezza 到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Netezza，然后选择 Netezza – Amazon Redshift。AWS SCT 显示 Netezza 转换到 Amazon Redshift 的所有可用设置。

AWS SCT 中的 Netezza 到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。
- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

Netezza 到 Amazon Redshift 的转换优化设置

要编辑 Netezza 到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Netezza，然后选择 Netezza – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示从 Netezza 转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中 Netezza 到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

要仅使用自动表优化，请在初始键选择策略中选择无。

- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：

  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

  您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

  此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

将 Oracle 数据仓库用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Oracle 数据仓库转换到 Amazon Redshift 或 Amazon Redshift 和 AWS Glue 的组合。

将 Oracle 数据仓库用作源的权限

下面列出了将 Oracle 数据仓库用作源所需的权限：

- 连接
- select_catalog_role
- select any dictionary

连接到作为源的 Oracle 数据仓库

使用 AWS Schema Conversion Tool 按照以下过程连接到 Oracle 数据仓库源数据库。

连接到 Oracle 数据仓库源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Oracle，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 Oracle 源数据仓库连接信息，请按照以下说明进行操作：

参数	操作

Type	选择连接到您的数据库的连接类型。根据类型，提供以下附加信息：
	• SID
	  • 服务器名称：源数据库服务器的域名系统（DNS）名称或 IP 地址。
	  • 服务器端口：键入用于连接到源数据库服务器的端口。

参数	操作

	• Oracle SID：Oracle 系统 ID (SID)。要查找 Oracle SID，请向您的 Oracle 数据库提交以下查询：
	
	`SELECT sys_context('userenv','instance_name') AS SID FROM dual;`
	• 服务名称
	  • Server name：键入源数据库服务器的 DNS 名称或 IP 地址。
	  • 服务器端口：键入用于连接到源数据库服务器的端口。
	  • 服务名称：要连接到的 Oracle 服务的名称。
	• TNS 别名
	  • TNS file path：包含透明网络底层 (TNS) 名称连接信息的文件的路径。
	  • TNS file path：用于连接到源数据库的此文件中的 TNS 别名。
	• TNS 连接标识符
	  • TNS 标识符：已注册 TNS 连接信息的标识符。

用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层 (SSL) 连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • SSL 身份验证：选择此选项使用 SSL 身份验证进行连接。 • 信任存储：包含证书的信任存储的位置。 • 密钥存储：包含私有密钥和证书的密钥存储的位置。如果选中 SSL 身份验证，此值是必填的，否则为可选值。

参数	操作
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
Oracle 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

Oracle 数据仓库到 Amazon Redshift 的转换设置

要编辑 Oracle 数据仓库到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上面的列表中选择 Oracle，然后选择 Oracle – Amazon Redshift。AWS SCT 显示 Oracle 数据仓库转换到 Amazon Redshift 的所有可用设置。

AWS SCT 中的 Oracle 数据仓库到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。
- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 在 Amazon Redshift 中将源表的分区迁移到单独的表。为此，请选择使用 UNION ALL 视图，然后输入 AWS SCT 可以为单个源表创建的最大目标表数。

  Amazon Redshift 不支持表分区。要模拟此行为并加快查询运行速度，AWS SCT 可以将源表的每个分区迁移到 Amazon Redshift 中的单独表中。然后，AWS SCT 创建一个包含所有这些表中的数据的视图。

  AWS SCT 自动确定源表中的分区数量。根据源表分区的类型，此数量可能会超过您可以应用于 Amazon Redshift 集群的表配额。为避免达到此配额，请输入 AWS SCT 可以为单个源表的分区创建的最大目标表数。默认选项为 368 个表，它表示一年中 366 天的分区和以及 `NO RANGE` 和 `UNKNOWN` 分区的两个表。
- 将数据类型格式化函数（例如 `TO_CHAR`、`TO_DATE` 和 `TO_NUMBER`）与 Amazon Redshift 不支持的日期时间格式元素进行转换。默认情况下，AWS SCT 使用扩展包函数来模拟转换后的代码中这些不支持的格式元素的用法。

  与 Amazon Redshift 中的日期时间格式字符串相比，Oracle 中的日期时间格式模型包含更多元素。如果源代码仅包含 Amazon Redshift 支持的日期时间格式元素，则无需在转换后的代码中使用扩展包函数。为避免在转换后的代码中使用扩展包函数，请选择 Oracle 代码中使用的日期类型格式元素与 Amazon Redshift 中的日期时间格式字符串相似。在这种情况下，转换后的代码运行更快。

  与 Amazon Redshift 中的数字格式字符串相比，Oracle 中的数字格式模型包含更多的元素。如果源代码仅包含 Amazon Redshift 支持的数字格式元素，则无需在转换后的代码中使用扩展包函数。为避免在转换后的代码中使用扩展包函数，请选择 Oracle 代码中使用的数字格式元素与 Amazon Redshift 中的数字格式字符串相似。在这种情况下，转换后的代码运行更快。
- 转换 Oracle `LEAD` 和 `LAG` 分析函数。默认情况下，AWS SCT 会为每个 `LEAD` 和 `LAG` 函数引发一个操作项。

  当源代码在这些函数中不使用默认偏移值时，AWS SCT 可以使用 `NVL` 函数模拟这些函数的用法。为此，请选择使用 NVL 函数模拟 Oracle LEAD 和 LAG 函数的行为。
- 要模拟 Amazon Redshift 集群中主键和唯一键的行为，请选择模拟主键和唯一键的行为。

  Amazon Redshift 不强制使用唯一键和主键，它们仅用于提供信息。如果在代码中使用这些约束，请确保 AWS SCT 在转换后的代码中模拟它们的行为。

- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

Oracle 数据仓库到 Amazon Redshift 的转换优化设置

要编辑 Oracle 数据仓库到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上面的列表中选择 Oracle，然后选择 Oracle – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示从 Oracle 数据仓库转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中 Oracle 数据仓库到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：
  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

  您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。
- 配置策略详细信息。

除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

- 对于排序键列数限制，在排序键中输入最大列数。
- 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
- 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
- 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

使用 Snowflake 作为 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Snowflake 转换到 Amazon Redshift。

将 Snowflake 用作源数据库的权限

您可以使用 SECURITYADMIN 角色和 SECURITYADMIN 会话上下文创建具有权限的角色并向该角色授予用户名。

以下示例创建了最低权限并将其授予 min_privs 用户。

```
create role role_name;
grant role role_name to role sysadmin;
grant usage on database db_name to role role_name;
grant usage on schema db_name.schema_name to role role_name;
grant usage on warehouse datawarehouse_name to role role_name;
grant monitor on database db_name to role role_name;
grant monitor on warehouse datawarehouse_name to role role_name;
grant select on all tables in schema db_name.schema_name to role role_name;
grant select on future tables in schema db_name.schema_name to role role_name;
grant select on all views in schema db_name.schema_name to role role_name;
grant select on future views in schema db_name.schema_name to role role_name;
grant select on all external tables in schema db_name.schema_name to role role_name;
grant select on future external tables in schema db_name.schema_name to role role_name;
grant usage on all sequences in schema db_name.schema_name to role role_name;
grant usage on future sequences in schema db_name.schema_name to role role_name;
grant usage on all functions in schema db_name.schema_name to role role_name;
grant usage on future functions in schema db_name.schema_name to role role_name;
```

```
grant usage on all procedures in schema db_name.schema_name to role role_name;
grant usage on future procedures in schema db_name.schema_name to role role_name;
create user min_privs password='real_user_password'
DEFAULT_ROLE = role_name DEFAULT_WAREHOUSE = 'datawarehouse_name';
grant role role_name to user min_privs;
```

在上述示例中，替换以下占位符：

- 将 *role_name* 替换为具有只读权限的角色名称。
- 将 *db_name* 替换为源数据库的名称。
- 将 *schema_name* 替换为源架构的名称。
- 将 *datawarehousename* 替换为所需数据仓库的名称。
- 将 *min_privs* 替换为具有最低权限的用户名。

DEFAULT_ROLE 和 DEFAULT_WAREHOUSE 参数是密钥敏感的。

配置对 Amazon S3 的安全访问权限

Amazon S3 存储桶的安全和访问权限管理策略允许 Snowflake 访问 S3 存储桶、从中读取数据并将数据写入存储桶。您可以使用 Snowflake STORAGE INTEGRATION 对象类型配置私有 Amazon S3 存储桶的安全访问权限。Snowflake 存储集成对象将身份验证责任委托给 Snowflake 身份和访问权限管理实体。

有关更多信息，请参阅 Snowflake 文档中的配置 Snowflake 存储集成以访问 Amazon S3。

连接到作为源的 Snowflake

使用 AWS Schema Conversion Tool 按照以下过程连接到源数据库。

连接到 Snowflake 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Snowflake，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

1. 对于 AWS 密钥，输入密钥名称。
2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。

- 要手动输入 Snowflake 源数据仓库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
数据库。	输入 Snowflake 数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您显式请求加密时，AWS SCT 才会以加密格式存储您的密码。
使用 SSL	如果要使用安全套接字层 (SSL) 连接到数据库，请选择此选项。在 SSL 选项卡上提供以下其他信息（如适用）： • 私有密钥路径：私有密钥的位置。 • 密码：私有密钥的密码。 有关 Snowflake 的 SSL 支持的更多信息，请参阅配置连接的安全选项。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。如果设置此选项，则可以存储数据库密码。这样做意味着您无需输入密码即可快速连接到数据库。
Snowflake 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。

6. 选择连接以连接到源数据库。

Snowflake 作为源的限制

将 Snowflake 作为 AWS SCT 源时，存在以下限制：

- 对象标识符在对象类型和父对象的上下文中必须是唯一的：

  数据库

  架构标识符在数据库中必须唯一。

  Schemas

  对象标识符（例如表和视图的标识符）在架构中必须是唯一的。

  表格/视图

  表中的列标识符必须唯一。
- 大型和 xlarge 集群节点类型的最大表数为 9,900。8xlarge 集群节点类型的最大表数为 100,000。限制包括在查询处理或系统维护期间由用户定义或由 Amazon Redshift 创建的临时表。有关更多信息，请参阅 Amazon Redshift 集群管理指南中的 Amazon Redshift 配额。
- 对于存储过程，输入和输出参数的最大数量为 32。

Snowflake 的源数据类型

接下来，您可以找到使用 AWS SCT 时支持的 Snowflake 源数据类型以及与 Amazon Redshift 目标的默认映射。

Snowflake 数据类型	Amazon Redshift 数据类型。
NUMBER	NUMERIC(38)
NUMBER(p)	如果 p =< 4，则为 SMALLINT 如果 p => 5 且 =< 9，则为 INTEGER 如果 p => 10 且 =< 18，则为 BIGINT 如果 p => 19 则为 NUMERIC(p)
NUMBER(p, 0)	如果 p =< 4，则为 SMALLINT

Snowflake 数据类型	Amazon Redshift 数据类型。
	如果 p => 5 且 =< 9，则为 INTEGER 如果 p => 10 且 =< 18，则为 BIGINT 如果 p => 19 则为：NUMERIC(p,0)
NUMBER(p, s)	如果 p => 1 且 =< 38，并且如果 s => 1 且 =< 37，那么 NUMERIC (p,s)
FLOAT	FLOAT
TEXT Unicode 字符最多 16,777,216 字节；每个字符最多 4 个字节。	VARCHAR(MAX)
TEXT(p) Unicode 字符最多 65,535 字节；每个字符最多 4 个字节。	如果 p =< 65,535 那么，VARCHAR (p)
TEXT(p) Unicode 字符最多 16,777,216 字节；每个字符最多 4 个字节。	如果 p => 65535 且 =< 16,777,216 则为 VARCHAR(MAX)
BINARY 单字节字符最多 8,388,608 字节；每个字符 1 字节。	VARCHAR(MAX)
BINARY(p) 单字节字符最多 65,535 字节；每个字符 1 字节。	VARCHAR(p)

Snowflake 数据类型	Amazon Redshift 数据类型。
BINARY(p) 单字节字符最多 8,388,608 字节；每个字符 1 字节。	VARCHAR(MAX)
BOOLEAN	BOOLEAN
DATE	DATE
TIME 介于 00:00:00 和 23:59:59.999999999 之间的时间值。	VARCHAR(18)
TIME(f) 介于 00:00:00 和 23:59:59.9(f) 之间的时间值。	VARCHAR(n) – 9 + dt-attr-1
TIMESTAMP_NTZ	TIMESTAMP
TIMESTAMP_TZ	TIMESTAMPTZ

Snowflake 到 Amazon Redshift 的转换设置

要编辑 Snowflake 到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Snowflake，然后选择 Snowflake – Amazon Redshift。AWS SCT 显示 Snowflake 到 Amazon Redshift 转换的所有可用设置。

AWS SCT 中的 Snowflake 到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。
- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

Snowflake 到 Amazon Redshift 的转换优化设置

要编辑 Snowflake 到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Snowflake，然后选择 Snowflake – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示从 Snowflake 转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中 Snowflake 到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：

- 使用元数据，忽略统计信息
- 忽略元数据，使用统计信息
- 使用元数据和统计信息

根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

- 配置策略详细信息。

除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

- 对于排序键列数限制，在排序键中输入最大列数。
- 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
- 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
- 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

将 Microsoft SQL Server 数据仓库作为 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Microsoft SQL Server 数据仓库转换到 Amazon Redshift 或 Amazon Redshift 和 AWS Glue 的组合。

将 Microsoft SQL Server 数据仓库用作源的权限

下面列出了将 Microsoft SQL Server 数据仓库用作源所需的权限：

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

在前面的示例中，将 *<source_schema>* 占位符替换为源 source_schema 的名称。

对要转换其架构的每个数据库重复这种授权。

此外，授予以下权限，并在主数据库上运行该授权：

- VIEW SERVER STATE

将 SQL Server 数据仓库作为源的限制

目前不支持将 Microsoft SQL Server 并行数据仓库（PDW）作为源。

连接到作为源的 SQL Server 数据仓库

使用 AWS Schema Conversion Tool 按照以下过程连接到 SQL Server 数据仓库源数据库。

连接到 SQL Server 数据仓库源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Microsoft SQL Server，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 Microsoft SQL Server 源数据仓库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名服务 (DNS) 名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
实例名称	输入 SQL Server 数据仓库的实例名称。

参数	操作
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层 (SSL) 连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 信任服务器证书：选择此选项以信任服务器证书。 • 信任存储：您在全局设置中设置的信任存储。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
SQL Server 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

SQL Server 数据仓库到 Amazon Redshift 的转换设置

要编辑 SQL Server 数据仓库到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Microsoft SQL Server，然后选择 Microsoft SQL Server – Amazon Redshift。AWS SCT 显示 SQL Server 数据仓库到 Amazon Redshift 转换的所有可用设置。

AWS SCT 中的 SQL Server 数据仓库到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 在 Amazon Redshift 中将源表的分区迁移到单独的表。为此，请选择使用 UNION ALL 视图，然后输入 AWS SCT 可以为单个源表创建的最大目标表数。

  Amazon Redshift 不支持表分区。要模拟此行为并加快查询运行速度，AWS SCT 可以将源表的每个分区迁移到 Amazon Redshift 中的单独表中。然后，AWS SCT 创建一个包含所有这些表中的数据的视图。

  AWS SCT 自动确定源表中的分区数量。根据源表分区的类型，此数量可能会超过您可以应用于 Amazon Redshift 集群的表配额。为避免达到此配额，请输入 AWS SCT 可以为单个源表的分区创建的最大目标表数。默认选项为 368 个表，它表示一年中 366 天的分区和以及 `NO RANGE` 和 `UNKNOWN` 分区的两个表。

- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

SQL Server 数据仓库到 Amazon Redshift 的转换优化设置

要编辑 SQL Server 数据仓库到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Microsoft SQL Server，然后选择 Microsoft SQL Server – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示从 SQL Server 数据仓库转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中 SQL Server 数据仓库到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：
  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

  您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。
- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。
  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

将 Teradata 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Teradata 转换到 Amazon Redshift 或 Amazon Redshift 和 AWS Glue 的组合。

将 Teradata 用作源的权限

下面列出了将 Teradata 用作源所需的权限：

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

在前面的示例中，将 *<source_database>* 占位符替换为源数据库的名称。

AWS SCT 需要具有 CREATE PROCEDURE 权限才能对源数据库中的所有过程执行 HELP PROCEDURE。AWS SCT 不会使用此权限在源 Teradata 数据库中创建任何新对象。

连接到作为源的 Teradata

使用 AWS Schema Conversion Tool 按照以下过程连接到 Teradata 源数据库。

连接到 Teradata 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Teradata，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

1. 对于 AWS 密钥，输入密钥名称。
2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。

- 要手动输入 Teradata 源数据库连接信息，请按照以下说明进行操作：

参数	操作
连接名称	输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
数据库。	输入 Teradata 数据库的名称。
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
加密数据	请选择此选项以加密与数据库交换的数据。如果您选择此选项，则使用端口号 443 在 AWS SCT 和 Teradata 数据库之间传输加密数据。
Teradata 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

Teradata 源使用 LDAP 身份验证

要为在 Windows 中运行 Microsoft Active Directory 的 Teradata 用户设置轻量目录访问协议 (LDAP) 身份验证，请使用以下过程。

在以下过程中，Active Directory 域为 `test.local.com`。Windows 服务器为 DC 并且使用了默认设置进行配置。以下脚本创建 `test_ldap` Active Directory 账户，并且该账户使用 `test_ldap` 密码。

为在 Windows 中运行 Microsoft Active Directory 的 Teradata 用户设置 LDAP 身份验证

1. 在 `/opt/teradata/tdat/tdgss/site` 目录中，编辑文件 `TdgssUserConfigFile.xml`。将 LDAP 部分更改为以下内容。

```
AuthorizationSupported="no"

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

2. 通过运行如下所示的配置来应用更改。

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

3. 通过运行以下命令来测试配置。

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

该输出值应该类似于以下内容。

```
LdapGroupBaseFQDN: dc=Test, dc=local, dc=com
LdapUserBaseFQDN: dc=Test, dc=local, dc=com
LdapSystemFQDN: dc= test, dc= local, dc=com
LdapServerName: DC.test.local.com
LdapServerPort: 389
LdapServerRealm: test.local.com
```

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

4. 使用以下命令重新启动 TPA。

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

5. 在 Teradata 数据库中创建与 Active Directory 中相同的用户，如下所示。

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

如果您在 Active Directory 中更改 LDAP 用户的用户密码，则应在 LDAP 模式下连接到 Teradata 时指定此新密码。在默认模式下，您仍必须使用 LDAP 用户名和任意密码连接 Teradata。

在源 Teradata 数据仓库中配置统计数据收集

要转换源 Teradata 数据仓库，AWS SCT 使用统计数据优化转换后的 Amazon Redshift 数据仓库。您可以在 AWS SCT 中收集统计数据或上传统计数据文件。有关更多信息，请参阅收集或上传统计数据。

要确保 AWS SCT 可以从您的数据仓库收集统计数据，请完成以下先决条件任务。

从 Teradata 数据仓库收集统计数据

1. 运行以下查询以重新收集数据仓库中所有表的统计信息。

```
collect summary statistics on table_name;
```

   在上述示例中，将 *table_name* 替换为源表名称。对您转换的每个表重复查询。

2. 运行以下查询以确定用于转换数据仓库的用户的账户字符串。

```
select * from dbc.accountinfo where username ='user_name'
```

3. 使用上一个示例中的账户字符串为特定用户开启查询日志记录。

```
BEGIN QUERY LOGGING WITH OBJECTS, SQL ON ALL ACCOUNT=('$M$BUSI$S$D$H');
```

或者，为所有数据库用户开启查询日志记录。

```
BEGIN QUERY LOGGING WITH SQL, OBJECTS LIMIT SQLTEXT=0 ON ALL;
```

收集完数据仓库统计数据后，关闭查询日志记录。为此，您可以使用以下代码示例。

```
end query logging with explain, objects, sql on all account=(' $M$BUSI$S$D$H');
```

在离线模下从源 Teradata 数据仓库收集统计数据

在 Teradata 数据仓库中配置统计数据收集后，即可在 AWS SCT 项目中收集统计数据。或者，您可以使用 Basic Teradata Query（BTEQ）脚本在离线模式下收集统计信息。然后，您可以将包含收集到的统计数据的文件上传到 AWS SCT 项目。有关更多信息，请参阅收集或上传统计数据。

在离线模式下从 Teradata 数据仓库收集统计数据

1. 创建包含以下内容的 `off-line_stats.bteq` 脚本。

```
.OS IF EXIST column-stats-tera.csv del /F column-stats-tera.csv
.OS IF EXIST table-stats-tera.csv del /F table-stats-tera.csv
.OS IF EXIST column-skew-script-tera.csv del /F column-skew-script-tera.csv
.OS IF EXIST column-skew-stats-tera.csv del /F column-skew-stats-tera.csv
.OS IF EXIST query-stats-tera.csv  del /F query-stats-tera.csv
.LOGON your_teradata_server/your_login, your_password
.EXPORT REPORT FILE = table-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

SELECT
    '"' || OREPLACE(COALESCE(c.DatabaseName, ''), '"', '""') || '";' ||
    '"' || OREPLACE(COALESCE(c.TableName, ''), '"', '""') || '";' ||
    '"' || TRIM(COALESCE(s.reference_count, '0')) || '";' ||
    '"' || TRIM(COALESCE(CAST(p.RowCount AS BIGINT), '0')) || '";' ||
```

```
    '"' || CAST(CAST(w.size_in_mb AS DECIMAL (38,1) FORMAT 'Z9.9') AS VARCHAR(38))
 || '";' ||
    '"' || TRIM(COALESCE(r.stat_fk_dep_count, '0')) || '";' ||
    '"' || CAST(CAST(current_timestamp(0) as timestamp(0) format 'YYYY-MM-
DDBHH:MI:SS') as VARCHAR(19)) || '"'
(TITLE
 '"database_name";"table_name";"reference_count";"row_count";"size_in_mb";"stat_fk_dep_coun
FROM (select databasename, tablename
        from DBC.tablesv
        where tablekind IN ('T','O')
        and databasename = 'your_database_name'
         ) c
left join
        (select DatabaseName, TableName, max(RowCount) RowCount
        from dbc.tableStatsv
        group by 1,2)p
on p.databasename = c.databasename
and p.tablename = c.tablename
left join
        (SELECT r.ChildDB as DatabaseName,
        r.ChildTable as TableName,
        COUNT(DISTINCT r.ParentTable) reference_count
        FROM DBC.All_RI_ChildrenV r
        GROUP BY r.ChildDB, r.ChildTable) s
on s.databasename = c.databasename
and s.tablename = c.tablename
left join
        (SELECT r.ParentDB as DatabaseName,
        r.ParentTable as TableName,
        COUNT(DISTINCT r.ChildTable) stat_fk_dep_count
        FROM DBC.All_RI_ParentsV r
        GROUP BY r.ParentDB, r.ParentTable) r
on r.databasename = c.databasename
and r.tablename = c.tablename
left join
        (select databasename, tablename,
        sum(currentperm)/1024/1024 as size_in_mb
        from dbc.TableSizeV
        group by 1,2) w
on w.databasename = c.databasename
and w.tablename = c.tablename
WHERE COALESCE(r.stat_fk_dep_count,0) + COALESCE(CAST(p.RowCount AS BIGINT),0) +
 COALESCE(s.reference_count,0) > 0;
```

```
.EXPORT RESET

.EXPORT REPORT FILE = column-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000
    '"' || TRIM(COALESCE(CAST(t2.card AS BIGINT), '0')) || '";' ||

SELECT
 '"' || OREPLACE(COALESCE(trim(tv.DatabaseName), ''), '"', '""') || '";' ||
     '"' || OREPLACE(COALESCE(trim(tv.TableName), ''), '"', '""') || '";' ||
 '"' || OREPLACE(COALESCE(trim(tv.columnname), ''), '"', '""') || '";' ||
                         '"' || TRIM(COALESCE(CAST(t2.card AS BIGINT), '0')) ||
 '";' ||

 '"' || CAST(current_timestamp AS VARCHAR(19)) || '"' (TITLE
 '"database_name";"table_name";"column_name";"cardinality";"current_ts"')
FROM dbc.columnsv tv
LEFT JOIN
(
 SELECT
  c.DatabaseName AS DATABASE_NAME,
  c.TABLENAME  AS TABLE_NAME,
  c.ColumnName AS COLUMN_NAME,
  c.UniqueValueCount AS CARD
 FROM dbc.tablestatsv c
 WHERE c.DatabaseName = 'your_database_name'
 AND c.RowCount <> 0
) t2
ON tv.DATABASENAME = t2.DATABASE_NAME
AND tv.TABLENAME = t2.TABLE_NAME
AND tv.COLUMNNAME = t2.COLUMN_NAME
WHERE t2.card > 0;

.EXPORT RESET

.EXPORT REPORT FILE = column-skew-script-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

SELECT
'SELECT CAST(''"' || TRIM(c.DatabaseName) || '";"' || TRIM(c.TABLENAME)  || '";"'
 || TRIM(c.COLUMNNAME) || '";"'' ||
TRIM(CAST(COALESCE(MAX(cnt) * 1.0 / SUM(cnt), 0) AS NUMBER FORMAT ''9.9999'')) ||
 ''";"'' ||
```

```
CAST(CURRENT_TIMESTAMP(0) AS VARCHAR(19)) || ''"'' AS VARCHAR(512))
AS """DATABASE_NAME"";""TABLE_NAME"";""COLUMN_NAME"";""SKEWED"";""CURRENT_TS"""
FROM(
SELECT COUNT(*) AS cnt
FROM "' || c.DATABASENAME || '"."' || c.TABLENAME ||
'" GROUP BY "' || c.COLUMNNAME || '") t' ||
 CASE WHEN ROW_NUMBER() OVER(PARTITION BY c.DATABASENAME
 ORDER BY c.TABLENAME DESC, c.COLUMNNAME DESC) <> 1
 THEN ' UNION ALL'
 ELSE ';' END (TITLE '--SKEWED--')
FROM dbc.columnsv c
INNER JOIN
(SELECT databasename, TABLENAME
FROM dbc.tablesv  WHERE tablekind = 'T'
AND  databasename = 'your_database_name') t
ON t.databasename = c.databasename
AND t.TABLENAME = c.TABLENAME
INNER JOIN
(SELECT databasename, TABLENAME, columnname FROM  dbc.indices GROUP BY 1,2,3
WHERE  TRANSLATE_CHK (databasename USING LATIN_TO_UNICODE) + TRANSLATE_CHK
 (TABLENAME USING LATIN_TO_UNICODE) + TRANSLATE_CHK (columnname USING
 LATIN_TO_UNICODE) = 0
) i
ON i.databasename = c.databasename
AND i.TABLENAME = c.TABLENAME
AND i.columnname = c.columnname
WHERE c.ColumnType NOT IN ('CO','JN','N','++','VA','UT','AN','XM','A1','BO')
ORDER BY c.TABLENAME, c.COLUMNNAME;

.EXPORT RESET

.EXPORT REPORT FILE = column-skew-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

.RUN FILE = column-skew-script-tera.csv

.EXPORT RESET

.EXPORT REPORT FILE = query-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 32000

SELECT
```

```
  '"' || RTRIM(CAST(SqlTextInfo AS VARCHAR(31900)), ';') || '";"' ||
  TRIM(QueryCount) || '";"' ||
  TRIM(QueryId) || '";"' ||
  TRIM(SqlRowNo) || '";"' ||
  TRIM(QueryParts) || '";"' ||
  CAST(CURRENT_TIMESTAMP(0) AS VARCHAR(19)) || '"'
(TITLE
 '"query_text";"query_count";"query_id";"sql_row_no";"query_parts";"current_ts"')
  FROM
  (
    SELECT  QueryId,  SqlTextInfo, SqlRowNo, QueryParts, QueryCount,
    SUM(QueryFirstRow) OVER (ORDER BY QueryCount DESC, QueryId ASC, SqlRowNo ASC
    ROWS UNBOUNDED PRECEDING) AS topN
    FROM
    (SELECT QueryId,  SqlTextInfo, SqlRowNo, QueryParts, QueryCount,
      CASE WHEN
      ROW_NUMBER() OVER (PARTITION BY QueryCount, SqlTextInfo ORDER BY QueryId,
 SqlRowNo) = 1 AND SqlRowNo = 1
    THEN 1 ELSE 0 END AS QueryFirstRow
    FROM (
      SELECT q.QueryId,  q.SqlTextInfo, q.SqlRowNo,
      MAX(q.SqlRowNo) OVER (PARTITION BY q.QueryId) QueryParts,
      COUNT(q.SqlTextInfo) OVER (PARTITION BY q.SqlTextInfo) QueryCount
      FROM DBC.dbqlsqltbl q
      INNER JOIN
      (
        SELECT QueryId
        FROM DBC.DBQLogTbl t
        WHERE TRIM(t.StatementType) IN ('SELECT')
        AND TRIM(t.AbortFlag) = '' AND t.ERRORCODE = 0
        AND  (CASE WHEN 'All users' IN ('All users') THEN 'All users' ELSE
 TRIM(t.USERNAME) END) IN ('All users') --user_name list
        AND t.StartTime > CURRENT_TIMESTAMP - INTERVAL '30' DAY
        GROUP BY 1
      ) t
      ON q.QueryId = t.QueryId
      INNER JOIN
      (
        SELECT QueryId
        FROM DBC.QryLogObjectsV
        WHERE ObjectDatabaseName = 'your_database_name'
        AND ObjectType = 'Tab'
        AND CollectTimeStamp > CURRENT_TIMESTAMP - INTERVAL '30' DAY
        GROUP BY 1
```

```
      ) r
      ON r.QueryId = t.QueryId
      WHERE q.CollectTimeStamp > CURRENT_TIMESTAMP - INTERVAL '30' DAY
    ) t
  ) t
  WHERE SqlTextInfo NOT LIKE '%";"%'
  ) q
  WHERE
  QueryParts >=1
  AND topN <= 50
  ORDER BY QueryCount DESC, QueryId, SqlRowNo
  QUALIFY COUNT(QueryId) OVER (PARTITION BY QueryId) = QueryParts;

.EXPORT RESET

.LOGOFF

.QUIT
```

2. 创建用于运行您在上一步中创建的 BTEQ 脚本的 td_run_bteq.bat 文件。此文件使用以下内容。

```
@echo off > off-line_stats1.bteq & setLocal enableDELAYedexpansion
@echo off > off-line_stats2.bteq & setLocal enableDELAYedexpansion

set old1=your_teradata_server
set new1=%1
set old2=your_login
set new2=%2
set old3=your_database_name
set new3=%3
set old4=your_password
set /p new4=Input %2 pass?

for /f "tokens=* delims= " %%a in (off-line_stats.bteq) do (
set str1=%%a
set str1=!str1:%old1%=%new1%!
>> off-line_stats1.bteq echo !str1!
)

for /f "tokens=* delims= " %%a in (off-line_stats1.bteq) do (
set str2=%%a
set str2=!str2:%old2%=%new2%!
```

```
>> off-line_stats2.bteq echo !str2!
)

type nul > off-line_stats1.bteq

for /f "tokens=* delims= " %%a in (off-line_stats2.bteq) do (
set str3=%%a
set str3=!str3:%old3%=%new3%!
>> off-line_stats1.bteq echo !str3!
)

type nul > off-line_stats2.bteq

for /f "tokens=* delims= " %%a in (off-line_stats1.bteq) do (
set str4=%%a
set str4=!str4:%old4%=%new4%!
>> off-line_stats2.bteq echo !str4!
)

del .\off-line_stats1.bteq

echo export starting...

bteq -c UTF8 < off-line_stats.bteq > metadata_export.log

pause
```

3. 创建用于运行您在上一步中创建的批处理文件的 runme.bat 文件。此文件使用以下内容。

```
.\td_run_bteq.bat ServerName UserName DatabaseName
```

在 runme.bat 文件中，将 *ServerName*、*UserName* 和 *DatabaseName* 替换为适用的值。

然后，运行 runme.bat 文件。转换到 Amazon Redshift 的每个数据仓库重复此步骤。

运行此脚本后，会收到三个文件，其中包含每个数据库的统计信息。您可以将这些文件上传到 AWS SCT 项目中。为此，请从项目的左侧面板中选择数据仓库，然后打开上下文 (右键单击) 菜单。选择上传统计数据。

Teradata 到 Amazon Redshift 的转换设置

要编辑 Teradata 到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Teradata，然后选择 Teradata – Amazon Redshift。AWS SCT 显示 Teradata 到 Amazon Redshift 转换的所有可用设置。

AWS SCT 中的 Teradata 到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 在 Amazon Redshift 中将源表的分区迁移到单独的表。为此，请选择使用 UNION ALL 视图，然后输入 AWS SCT 可以为单个源表创建的最大目标表数。

  Amazon Redshift 不支持表分区。要模拟此行为并加快查询运行速度，AWS SCT 可以将源表的每个分区迁移到 Amazon Redshift 中的单独表中。然后，AWS SCT 创建一个包含所有这些表中的数据的视图。

  AWS SCT 自动确定源表中的分区数量。根据源表分区的类型，此数量可能会超过您可以应用于 Amazon Redshift 集群的表配额。为避免达到此配额，请输入 AWS SCT 可以为单个源表的分区创建的最大目标表数。默认选项为 368 个表，它表示一年中 366 天的分区和以及 `NO RANGE` 和 `UNKNOWN` 分区的两个表。

- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。
- 要在 SELECT * 语句转换后的代码中使用显式列列表，请选择使用显式列声明。
- 要模拟 Amazon Redshift 集群中主键和唯一键的行为，请选择模拟主键和唯一键的行为。

  Amazon Redshift 不强制使用唯一键和主键，它们仅用于提供信息。如果在代码中使用这些约束，请确保 AWS SCT 在转换后的代码中模拟它们的行为。
- 确保目标 Amazon Redshift 表中的数据唯一性。为此，请选择模拟 SET 表的行为。

  Teradata 使用 SET 语法元素作为默认选项创建表。您不能在 SET 表中添加重复的行。如果源代码不使用此唯一性约束，请关闭此选项。在这种情况下，转换后的代码运行更快。

  如果源代码使用表中的 SET 选项作为唯一性约束，请启用此选项。在这种情况下，AWS SCT 在转换后的代码中重写 INSERT..SELECT 语句以模拟源数据库的行为。

Teradata 到 Amazon Redshift 的转换优化设置

要编辑 Teradata 到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Teradata，然后选择 Teradata – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示从 Teradata 转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中 Teradata 到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：

- 使用元数据，忽略统计信息
- 忽略元数据，使用统计信息
- 使用元数据和统计信息

根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1-100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

  此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

将 Vertica 用作 AWS SCT 的源

您可以使用 AWS SCT 将架构、代码对象和应用程序代码从 Vertica 转换到 Amazon Redshift。

将 Vertica 用作源的权限

下面列出了将 Vertica 用作源所需的权限：

- USAGE ON SCHEMA *<schema_name>*
- USAGE ON SCHEMA PUBLIC
- SELECT ON ALL TABLES IN SCHEMA *<schema_name>*
- SELECT ON ALL SEQUENCES IN SCHEMA *<schema_name>*

- EXECUTE ON ALL FUNCTIONS IN SCHEMA *<schema_name>*
- EXECUTE ON PROCEDURE *<schema_name.procedure_name(procedure_signature)>*

在上述示例中，替换以下占位符：

- 将 *schema_name* 替换为源架构的名称。
- 将 *procedure_name* 替换为源过程的名称。对正在转换的每个过程重复授权。
- 将 *procedure_signature* 替换为以逗号分隔的过程参数类型列表。

连接到作为源的 Vertica

使用 AWS Schema Conversion Tool 按照以下过程连接到 Vertica 源数据库。

连接到 Vertica 源数据库

1. 在 AWS Schema Conversion Tool 中，选择添加源。
2. 选择 Vertica，然后选择下一步。

   此时显示添加源对话框。
3. 对于连接名称，输入数据库的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 使用来自 AWS Secrets Manager 的数据库凭证或手动输入：

   - 要使用 Secrets Manager 中的数据库凭证，请按照以下说明进行操作：

     1. 对于 AWS 密钥，输入密钥名称。
     2. 选择填充可使用 Secrets Manager 中的数据库凭证自动填写数据库连接对话框中的所有值。

     有关使用 Secrets Manager 中的数据库凭证的信息，请参阅使用 AWS Secrets Manager。
   - 要手动输入 Vertica 源数据库连接信息，请按照以下说明进行操作：

参数	操作
服务器名称	输入源数据库服务器的域名系统（DNS）名称或 IP 地址。
服务器端口	输入用于连接到源数据库服务器的端口。
数据库。	输入 Vertica 数据库的名称。

参数	操作
用户名和密码	输入数据库凭证，以便连接到源数据库服务器。 仅当您选择连接到项目中的数据库时，AWS SCT 才使用密码连接到源数据库。为了避免泄露源数据库的密码，AWS SCT 不会默认存储该密码。如果您关闭了 AWS SCT 项目并重新打开它，系统会根据需要提示您输入用于连接到源数据库的密码。
使用 SSL	选择此选项以使用安全套接字层 (SSL) 连接到数据库。在 SSL 选项卡上提供以下其他信息（如适用）： • 验证服务器证书：选择此选项以使用信任存储验证服务器证书。 • 信任存储：您在全局设置中设置的信任存储。 • 密钥存储：您在全局设置中设置的密钥存储。
存储密码	AWS SCT 将创建一个安全文件库，用于存储 SSL 证书和数据库密码。启用此选项，可存储数据库密码，且无需输入密码可快速连接到数据库。
Vertica 驱动程序路径	输入用于连接到源数据库的驱动程序的路径。有关更多信息，请参阅下载所需的数据库驱动程序。 如果您将驱动程序路径存储在全局项目设置中，则驱动程序路径不会显示在连接对话框中。有关更多信息，请参阅在全局设置中存储驱动程序路径。

5. 选择测试连接以验证 AWS SCT 是否可以连接到源数据库。
6. 选择连接以连接到源数据库。

Vertica 到 Amazon Redshift 的转换设置

要编辑 Vertica 到 Amazon Redshift 的转换设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Vertica，然后选择 Vertica – Amazon Redshift。AWS SCT 显示 Vertica 到 Amazon Redshift 转换的所有可用设置。

AWS SCT 中的 Vertica 到 Amazon Redshift 转换设置包括以下各项的选项：

- 限制转换后的代码中操作项的注释数量。

  对于在转换后的代码中为所选严重性级别及更高级别的操作项添加注释，请选择操作项的严重性。AWS SCT 会在转换后的代码中为所选严重性级别及更高级别的操作项添加注释。

  例如，要最大限度地减少转换后的代码中的注释数量，请选择仅错误。要在转换后的代码中包含所有操作项的注释，请选择所有消息。

- 设置 AWS SCT 可以应用于目标 Amazon Redshift 集群的最大表数。

  对于目标 Amazon Redshift 集群的最大表数，请选择 AWS SCT 可以应用于 Amazon Redshift 集群的表数量。

  Amazon Redshift 具有限制了不同集群节点类型使用表数的配额。如果选择自动，则 AWS SCT 会根据节点类型确定要应用于目标 Amazon Redshift 集群的表数量。或者，手动选择值。有关更多信息，请参阅《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配额和限制。

  AWS SCT 转换所有源表，即使表的数量超过了您的 Amazon Redshift 集群所能存储的容量也是如此。AWS SCT 将转换后的代码存储在项目中，并且不会将其应用于目标数据库。如果应用转换后的代码时达到了 Amazon Redshift 集群的表配额，则 AWS SCT 会显示一条警告消息。此外，AWS SCT 还要将表应用于目标 Amazon Redshift 集群，直到表的数量达到上限。

- 在 Amazon Redshift 中将源表的分区迁移到单独的表。为此，请选择使用 UNION ALL 视图，然后输入 AWS SCT 可以为单个源表创建的最大目标表数。

  Amazon Redshift 不支持表分区。要模拟此行为并加快查询运行速度，AWS SCT 可以将源表的每个分区迁移到 Amazon Redshift 中的单独表中。然后，AWS SCT 创建一个包含所有这些表中的数据的视图。

  AWS SCT 自动确定源表中的分区数量。根据源表分区的类型，此数量可能会超过您可以应用于 Amazon Redshift 集群的表配额。为避免达到此配额，请输入 AWS SCT 可以为单个源表的分区创建的最大目标表数。默认选项为 368 个表，它表示一年中 366 天的分区和以及 `NO RANGE` 和 `UNKNOWN` 分区的两个表。

- 对 Amazon Redshift 表列应用压缩。为此，请选择使用压缩编码。

  AWS SCT 使用默认 Amazon Redshift 算法自动为列分配压缩编码。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。

  默认情况下，Amazon Redshift 不对定义为排序键和分配键的列应用压缩。您可以更改此行为并对这些列进行压缩。为此，请选择为 KEY 列使用压缩编码。只有选择使用压缩编码选项时，才能选择此选项。

Vertica 到 Amazon Redshift 的转换优化设置

要编辑 Vertica 到 Amazon Redshift 的转换优化设置，请在 AWS SCT 中选择设置，然后选择转换设置。从上方的列表中选择 Vertica，然后选择 Vertica – Amazon Redshift。在左窗格中，选择优化策略。AWS SCT 显示从 Vertica 转换到 Amazon Redshift 的转换优化设置。

AWS SCT 中 Vertica 到 Amazon Redshift 的转换优化设置包括以下各项的选项：

- 使用自动表优化。为此，请选择使用 Amazon Redshift 自动调整表格。

  自动表优化是 Amazon Redshift 中的一种自我调整过程，可自动优化表的设计。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的使用自动表优化。

  要仅使用自动表优化，请在初始键选择策略中选择无。
- 使用策略选择排序键和分配键。

  您可以使用 Amazon Redshift 元数据、统计信息或这两个选项选择排序键和分配键。对于优化策略选项卡上的初始键选择策略，请选择以下选项之一：

  - 使用元数据，忽略统计信息
  - 忽略元数据，使用统计信息
  - 使用元数据和统计信息

  根据您选择的选项，您可以选择优化策略。然后，请为每种策略输入值（0–100）。这些值定义了每种策略的权重。AWS SCT 使用这些权重值定义每条规则如何影响分布键和排序键的选择。默认值基于 AWS 迁移最佳实践。

  您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格的最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。
- 配置策略详细信息。

  除了定义每种优化策略的权重外，您还可以配置优化设置。为此，请选择转换优化。

  - 对于排序键列数限制，在排序键中输入最大列数。
  - 在偏斜阈值中，输入列偏斜值的百分比（0–100）。AWS SCT 从分配键的候选列表中排除倾斜值大于阈值的列。AWS SCT 将列的偏斜值定义为最常见值的出现次数与记录总数的百分比。
  - 对于查询历史表中的前 N 个查询，请输入要分析的最常用查询的数量（1–100）。
  - 在选择统计数据用户中，选择要分析查询统计数据的数据库用户。

此外，在优化策略选项卡上，您可以为查找小型表策略定义小型表的大小。在最小表格行数和最大表格行数中，输入表格中最小和最大行数以将其定义为小型表。AWS SCT 将 ALL 分配方式应用于小型表。在这种情况下，向每个节点分配整个表的副本。

在 AWS SCT 中创建映射规则

您可以在单个 AWS SCT 项目中添加多个源数据库和目标数据库。将多个数据库迁移到不同的目标平台可以简化项目管理。

创建新项目并添加源数据库和目标数据库后，创建映射规则。AWS SCT 需要至少一条映射规则才能创建迁移评估报告并转换数据库架构。

映射规则描述了源目标对，其中包括源数据库架构或源数据库以及目标数据库平台。您可以在单个 AWS SCT 项目中创建多个映射规则。使用映射规则将每个源数据库架构转换到正确的目标数据库平台。

要在转换后的代码中更改架构的名称，请设置迁移规则。例如，使用迁移规则，您可以重命名架构、为对象名称添加前缀、更改列排序规则或更改数据类型。要将这些更改应用于转换后的代码，请确保在转换源架构之前创建迁移规则。有关更多信息，请参阅 创建迁移规则。

您只能为支持的数据库转换对创建映射规则。有关支持的转换对列表信息，请参阅 AWS SCT 的源。

如果打开保存在版本 1.0.655 或更早版本中的 AWS SCT 的项目，则 AWS SCT 会自动为所有源数据库架构创建到目标数据库平台的映射规则。要添加其他目标数据库平台，请删除现有的映射规则，然后创建新的映射规则。

主题

添加新映射规则

您可以在单个项目中创建多个映射规则。AWS SCT 会将映射规则作为项目的一部分保存。打开项目，使用以下过程创建新的映射规则。

创建映射规则

1. 在视图菜单上，选择映射视图。

2. 在左侧面板中，选择要添加到映射规则的架构或数据库。
3. 在右侧面板中，为所选源架构或数据库选择目标数据库平台。

   您可以选择虚拟数据库平台作为目标。有关更多信息，请参阅使用虚拟目标。
4. 选择创建映射。

   AWS SCT 将此新映射规则添加到服务器映射列表中。

添加所有转换对的映射规则。要创建评估报告或转换数据库架构，请在视图菜单上选择主视图。

AWS SCT 以粗体突出显示属于映射规则的所有架构对象。

管理映射规则

您可以筛选或删除现有的映射规则，并在 AWS Schema Conversion Tool（AWS SCT）项目中添加新的映射规则。

为整个源数据库创建映射规则时，AWS SCT 会为每个源数据库架构创建一个映射规则。对于涉及数十个架构甚至数据库的项目，可能很难确定某个架构的目标。要快速找到架构的映射规则，请在 AWS SCT 中使用以下一个或多个筛选选项。

筛选映射规则

1. 在视图菜单上，选择映射视图。
2. 在源数据库中，选择您的源数据库。

   筛选默认值为全部，这表示 AWS SCT 显示所有源数据库的映射规则。
3. 在源架构中，输入源架构名称。使用百分比（%）作为通配符来替换架构名称中任意数量的任何符号。

   筛选默认值为%通配符，这表示 AWS SCT 显示所有源数据库架构名称的映射规则。
4. 对于有迁移规则，选择是以显示创建数据迁移规则的映射规则。选择否可显示没有数据迁移规则的映射规则。有关更多信息，请参阅在中创建数据迁移规则 AWS SCT。

   筛选默认值为全部，这表示 AWS SCT 显示所有映射规则。
5. 对于目标服务器，选择您的目标数据库。

   筛选默认值为全部，这表示 AWS SCT 显示所有目标数据库的映射规则。

打开项目，使用以下过程删除映射规则。有关添加映射规则的更多信息，请参阅 添加新映射规则。

删除映射规则

1. 在视图菜单上，选择映射视图。
2. 对于服务器映射，请选择要删除的映射规则。
3. 选择删除选定的映射。

   AWS SCT 删除选定的映射规则。

使用虚拟目标

您可以看到 AWS SCT 如何将源数据库架构转换为任何支持的目标数据库平台。为此，您无需连接到现有目标数据库。相反，您可以在创建映射规则时在右侧面板中选择虚拟目标数据库平台。有关更多信息，请参阅添加新映射规则。确保展开右侧面板中的服务器、NoSQL 集群和 ETL 节点，以查看虚拟目标数据库平台列表。

AWS SCT 支持以下虚拟目标数据库平台：

- Amazon Aurora MySQL 兼容版
- Amazon Aurora PostgreSQL 兼容版
- Amazon DynamoDB
- Amazon Redshift
- Amazon Redshift 和 AWS Glue
- AWS Glue
- AWS Glue Studio
- 适用于 Aurora PostgreSQL 的 Babelfish
- MariaDB
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL

如果使用适用于 Aurora PostgreSQL 的 Babelfish 作为目标数据库平台，则只能创建数据库迁移评估报告。有关更多信息，请参阅the section called “迁移评估报告”。

如果使用虚拟目标数据库平台，则可以将转换后的代码保存到文件中。有关更多信息，请参阅the section called “保存转换后的架构”。

在单个 AWS SCT 项目中使用多个服务器的限制

在单个 AWS SCT 项目中使用多个服务器转换架构时，存在以下限制：

- 您只能将同一台服务器添加到项目中一次。
- 您不能将服务器架构映射到特定的目标架构，只能映射到目标服务器。AWS SCT 在转换过程中创建目标架构。
- 您无法将较低级别的源对象映射到目标服务器。
- 在一个项目中，您只能将一个源架构映射到一个目标服务器。
- 确保将源映射到目标服务器以创建评估报告、转换架构或提取数据。

创建转换报告

当您规划数据库转换时，创建一些报告来帮助您理解所涉及的内容会很有用。可以使用 AWS Schema Conversion Tool 创建报告。

可以使用 AWS SCT 创建数据库迁移评估报告。借助此报告，您将大致了解您的架构转换任务以及无法自动转换为目标数据库的项目的详细信息。您可以使用此报告评估 AWS SCT 完成项目的程度以及完成转换所需的其他操作。要创建评估报告，请使用 AWS SCT 数据库上下文（右键单击）菜单中的创建报告。

主题

使用 AWS SCT 创建评估报告

AWS Schema Conversion Tool 的一个重要部分是其生成的评估报告，用于估算架构转换的复杂性。该数据库迁移评估报告汇总了所有架构转换任务，针对无法转换为目标数据库实例的数据库引擎的架构，还详细介绍了其操作项。您可以在应用程序中查看报告，也可以将其导出为逗号分隔值（CSV）或 PDF 文件。

如果您在单个项目中添加多个源数据库和目标数据库，AWS SCT 将所有转换对的报告汇总到一份数据库迁移评估报告中。

您可以使用虚拟目标数据库平台生成评估报告，并了解迁移到所选数据库平台的复杂性。在这种情况下，不需要连接到目标数据库平台。例如，你可以使用适用于 Aurora PostgreSQL 的 Babelfish 作为虚拟目标数据库平台来创建数据库迁移评估报告。有关虚拟目标数据库平台的更多信息，请参阅 the section called “虚拟目标”。

该迁移评估报告包括以下内容：

- 执行摘要
- 许可评估
- 云支持，指示在目标数据库中不可用的任何源数据库功能。
- 建议，包括服务器对象的转换、备份建议和链接的服务器更改

此外，该报告还包含：对于无法自动转换的部分，在您的目标数据库实例中编写等效代码所需工作量的估算。

如果您使用 AWS SCT 将现有架构迁移到 Amazon RDS 数据库实例，该报告可帮助您分析关于迁移到 AWS 云和更改许可证类型的要求。

主题

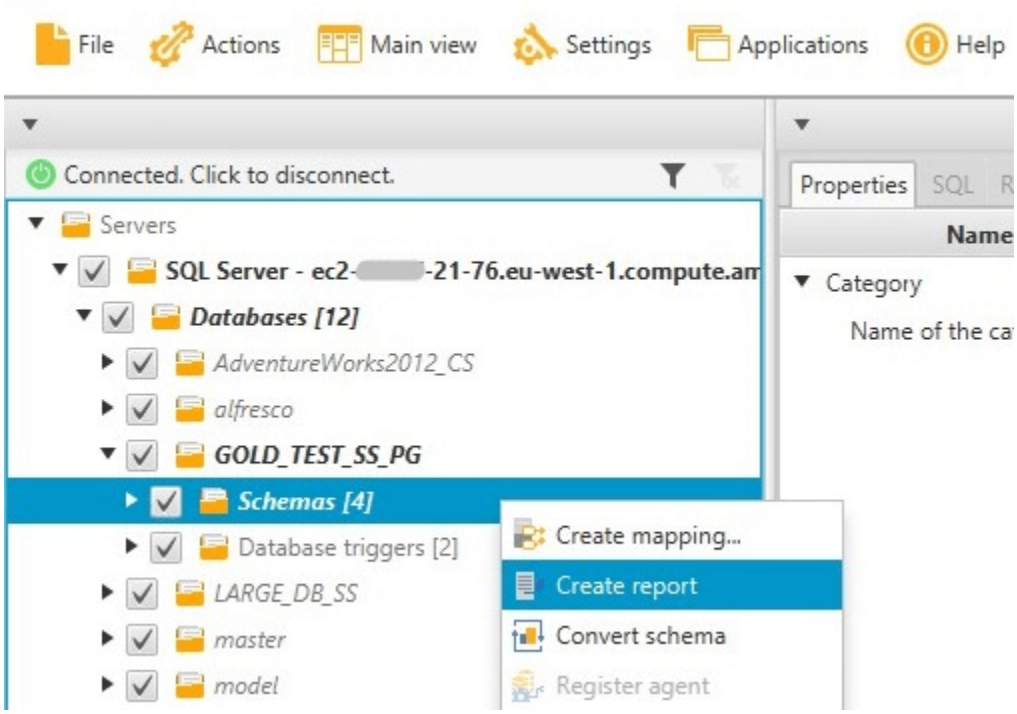
创建数据库迁移评估报告

使用以下过程创建数据库迁移评估报告。

创建数据库迁移评估报告

1. 确保您创建了源数据库架构映射规则，以便为其创建评估报告。有关更多信息，请参阅添加新映射规则。
2. 在视图菜单上，选择主视图。
3. 在显示源数据库架构的左侧面板中，选择要为其创建评估报告的架构对象。如需在报告中包含多个数据库架构，请选择父节点，例如架构。

   确保选中了要创建评估报告的所有架构对象的复选框。
4. 打开该对象的上下文 (右键单击) 菜单，然后选择创建报告。



查看评估报告

在创建评估报告后，评估报告视图将打开，其中显示以下选项卡：

- 摘要
- Action Items (操作项)

Summary (摘要) 选项卡显示已自动转换或未转换的项目。

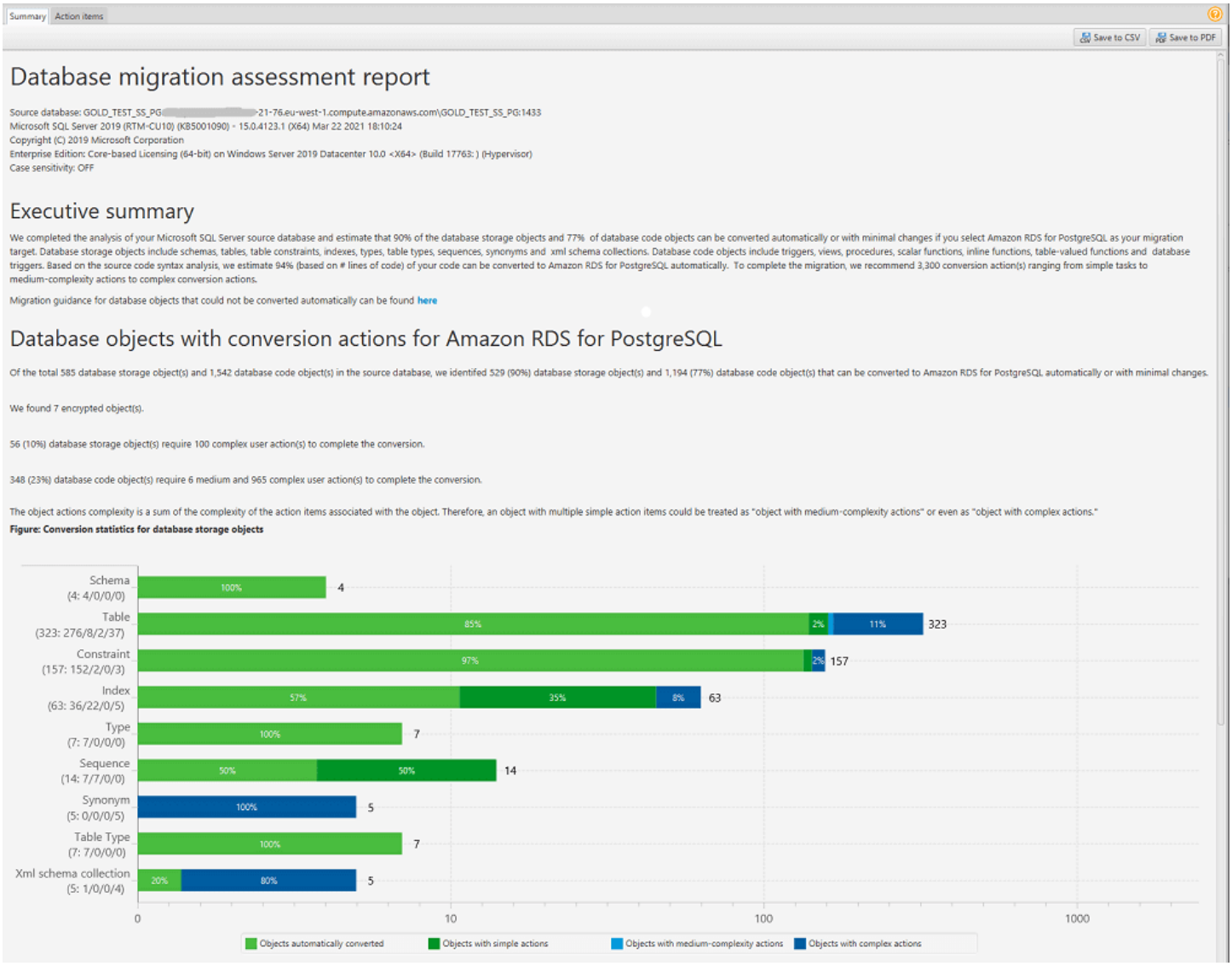
Action Items (操作项) 选项卡显示无法自动转换的项目，以及有关如何处理它们的建议。

主题

- 评估报告摘要
- 评估报告操作项
- 评估报告警告消息

评估报告摘要

Summary 选项卡显示了来自数据库迁移评估报告的摘要信息。它显示了已自动转换的项目和未自动转换的项目。

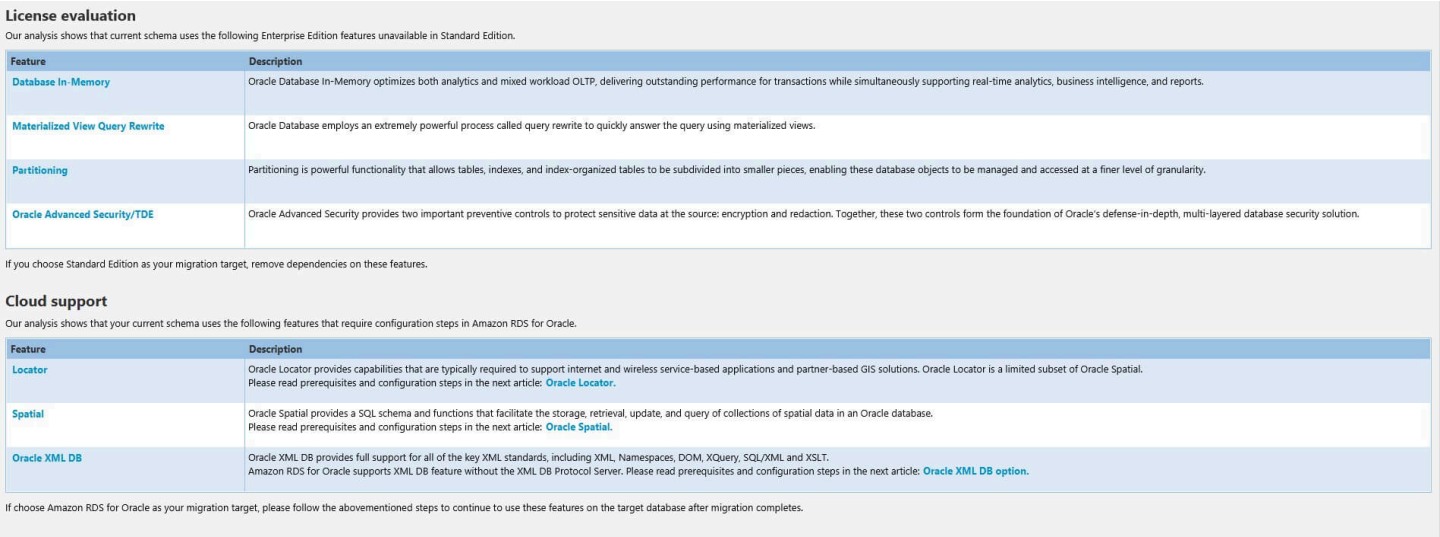


对于无法自动转换为目标数据库引擎的架构项目，摘要包含了在您的目标数据库实例中创建与源数据库中的架构项目等效的架构项目所需的工作量的估算。

该报告将转换这些架构项目的估算时间划分为以下类别：

- 简单：可在 2 小时之内完成的操作。
- 中等：可在 2 到 6 小时之间完成的较复杂的操作。
- 大量：需要 6 小时以上才能完成的非常复杂的操作。

许可证评估和云支持部分包含关于将现有的本地数据库架构移到运行相同引擎的 Amazon RDS 数据库实例的信息。例如，如果您希望更改许可证类型，报告的此部分将告诉您应从当前数据库中删除哪些功能。



License evaluation

Our analysis shows that current schema uses the following Enterprise Edition features unavailable in Standard Edition.

Feature	Description
Database In-Memory	Oracle Database In-Memory optimizes both analytics and mixed workload OLTP, delivering outstanding performance for transactions while simultaneously supporting real-time analytics, business intelligence, and reports.
Materialized View Query Rewrite	Oracle Database employs an extremely powerful process called query rewrite to quickly answer the query using materialized views.
Partitioning	Partitioning is powerful functionality that allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity.
Oracle Advanced Security/TDE	Oracle Advanced Security provides two important preventive controls to protect sensitive data at the source: encryption and redaction. Together, these two controls form the foundation of Oracle's defense-in-depth, multi-layered database security solution.

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

Cloud support

Our analysis shows that your current schema uses the following features that require configuration steps in Amazon RDS for Oracle.

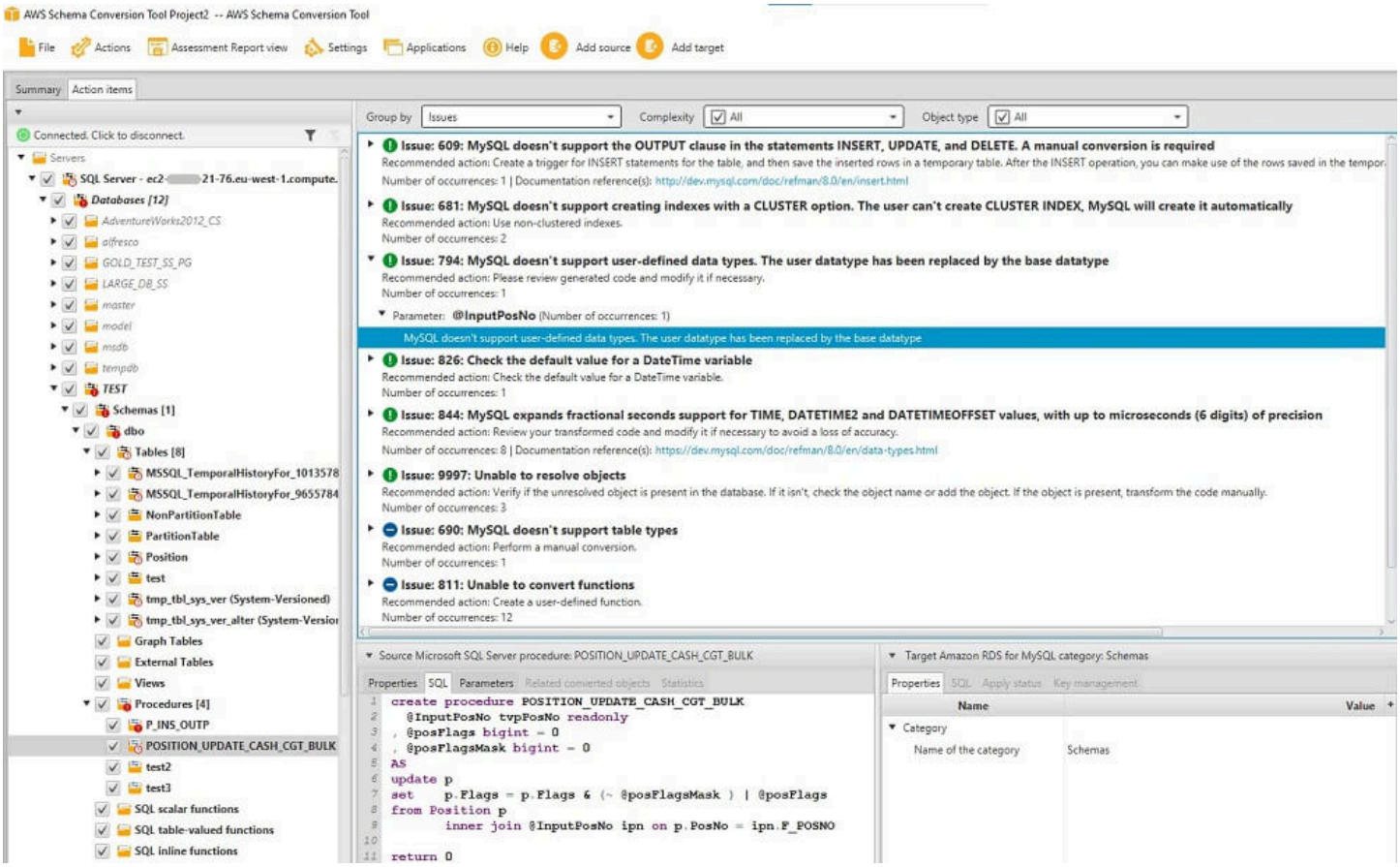
Feature	Description
Locator	Oracle Locator provides capabilities that are typically required to support internet and wireless service-based applications and partner-based GIS solutions. Oracle Locator is a limited subset of Oracle Spatial. Please read prerequisites and configuration steps in the next article: Oracle Locator.
Spatial	Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial data in an Oracle database. Please read prerequisites and configuration steps in the next article: Oracle Spatial.
Oracle XML DB	Oracle XML DB provides full support for all of the key XML standards, including XML, Namespaces, DOM, XQuery, SQL/XML and XSLT. Amazon RDS for Oracle supports XML DB feature without the XML DB Protocol Server. Please read prerequisites and configuration steps in the next article: Oracle XML DB option.

If choose Amazon RDS for Oracle as your migration target, please follow the abovementioned steps to continue to use these features on the target database after migration completes.

评估报告操作项

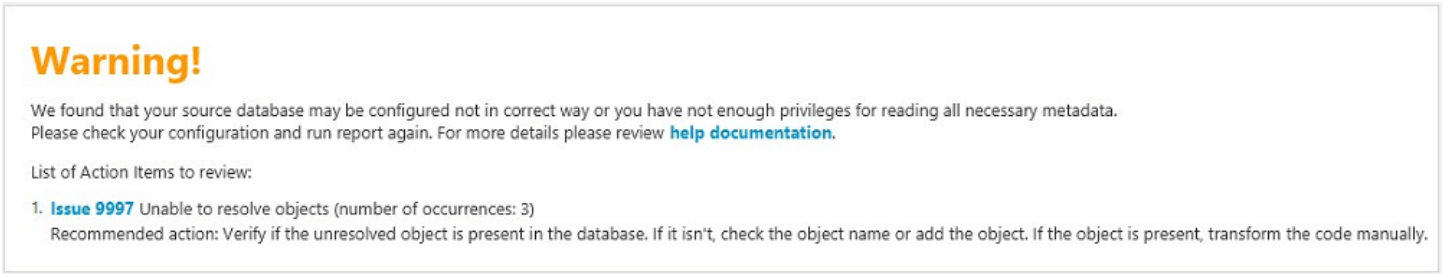
评估报告视图还包含 Action Items 选项卡。此选项卡包含无法自动转换为目标 Amazon RDS 数据库实例的数据库引擎的项列表。如果从列表中选择一个操作项，AWS SCT 会突出显示您架构中该操作项适用的项。

该报告还包含有关如何手动转换架构项目的建议。例如，评估运行后，数据库/架构的详细报告会显示设计和实施操作项转换的建议所需的工作量。有关如何处理手动转换的更多信息，请参阅在 AWS SCT 中处理手动转换。



评估报告警告消息

要评估转换为其他数据库引擎的复杂性，AWS SCT 需要访问源数据库中的对象。当 SCT 因为扫描过程中遇到问题而无法执行评估时，会发出警告消息，表明总体转换百分比已降低。



以下是在扫描过程中 AWS SCT 可能遇到问题的原因：

- 连接到数据库的用户帐户无权访问所有所需对象。
- 数据库中不再存在架构中引用的对象。
- SCT 正在尝试评估加密的对象。

有关 SCT 所需的数据库安全权限的更多信息，请参阅本指南中相应的源数据库部分的 AWS SCT 的源。

保存评估报告

创建数据库迁移评测报告后，您可以将数据库迁移评估报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。

将数据库迁移评估报告另存为 PDF 文件。

1. 在顶部菜单中，选择视图，然后选择评估报告视图。
2. 选择 Summary 选项卡。
3. 选择右上角的保存为 PDF。

将数据库迁移评测报告保存为 CSV 文件

1. 在顶部菜单中，选择视图，然后选择评估报告视图。
2. 选择 Summary 选项卡。
3. 选择右上角的保存为 CSV。

PDF 文件包含摘要和操作项信息，如以下示例所示。

Database objects with conversion actions for Amazon RDS for PostgreSQL

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

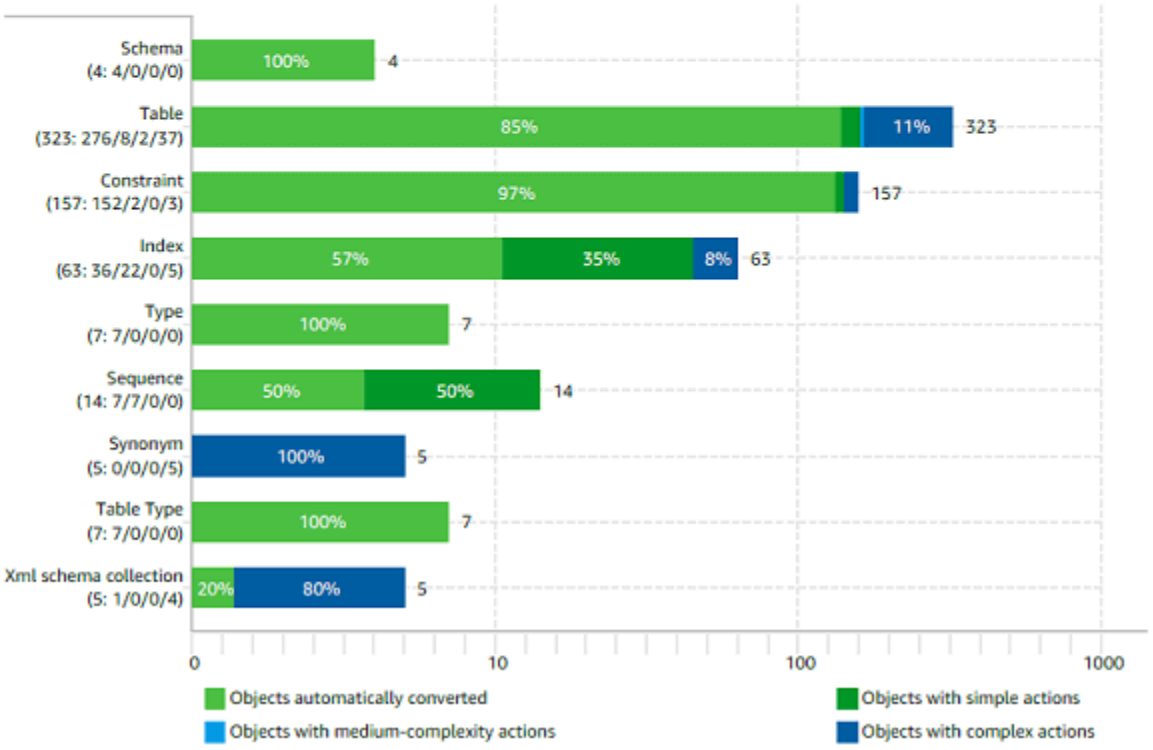
We found 7 encrypted object(s).

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

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

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

Figure: Conversion statistics for database storage objects



选择保存为 CSV 选项时，AWS SCT 会创建三个 CSV 文件。

第一个 CSV 文件包含有关操作项的以下信息：

- 类别
- 出现次数：项目的文件名、行号和位置
- 操作项编号
- Subject
- 组

- 描述
- 文档参考
- 推荐操作
- 估计的复杂性

第二个 CSV 文件的名称中包含 Action_Items_Summary 后缀，并包含有关所有操作项出现次数的信息。

在以下示例中，学习曲线工作量列中的值表示设计转换每个操作项的方法所需的工作量。转换操作项出现次数的工作量列中的值表示按照设计的方法转换每个操作项所需的工作量。用于表示所需工作量的值基于加权量表，范围从低（最小）到高（最大）。

Schema	Action item	Number of occurrences	Learning curve efforts	Efforts to convert an occurrence of the action item
TEST.dbo	609	1	8	0.3
TEST.dbo	681	2	0.1	0.1
TEST.dbo	690	1	40	40
TEST.dbo	794	1	0	0.01
TEST.dbo	811	12	40	8
TEST.dbo	826	1	0	0.1
TEST.dbo	844	8	8	0.5
TEST.dbo	9997	3	0	0.3

第三个 CSV 文件的名称中包含 Summary，并包含以下摘要：

- 类别
- 对象数量
- 已自动转换的对象
- 需要简单操作的对象
- 需要中等复杂操作的对象
- 需要复杂操作的对象
- 代码总行数

配置评估报告

您可以配置 AWS SCT 包含在评估报告中的详细信息数量。

配置数据库迁移评估报告

1. 在设置菜单上，选择全局设置，然后选择评估报告。

2. 对于操作项出现次数，请选择仅前五个问题，以限制评估报告中单一类型的操作项的数量。选择所有问题，在评估报告中包括每种类型的所有操作项。
3. 对于 SQL 脚本分析的文件，请选择列出不超过 ***X*** 文件，将评估报告中的 SQL 脚本文件数限制为 *X*。输入文件数量。选择列出所有已分析文件，将所有 SQL 脚本文件包括在评估报告中。
4. 选择保存后打开报告，以便在保存数据库迁移评估报告的本地副本后自动打开该文件。有关更多信息，请参阅

创建数据库迁移评测报告后，您可以将数据库迁移评估报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。

将数据库迁移评估报告另存为 PDF 文件。

1. 在顶部菜单中，选择视图，然后选择评估报告视图。
2. 选择 Summary 选项卡。
3. 选择右上角的保存为 PDF。

将数据库迁移评测报告保存为 CSV 文件

1. 在顶部菜单中，选择视图，然后选择评估报告视图。
2. 选择 Summary 选项卡。
3. 选择右上角的保存为 CSV。

PDF 文件包含摘要和操作项信息，如以下示例所示。

Database objects with conversion actions for Amazon RDS for PostgreSQL

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

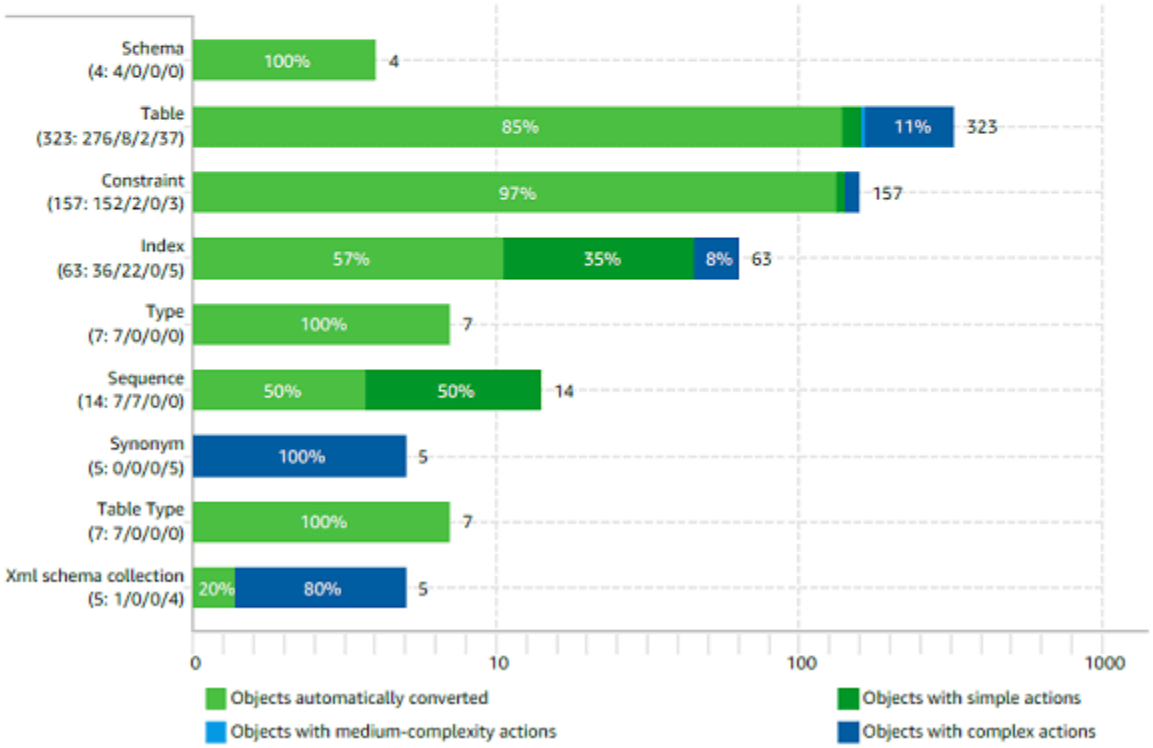
We found 7 encrypted object(s).

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

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

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

Figure: Conversion statistics for database storage objects



选择保存为 CSV 选项时，AWS SCT 会创建三个 CSV 文件。

第一个 CSV 文件包含有关操作项的以下信息：

- 类别
- 出现次数：项目的文件名、行号和位置
- 操作项编号
- Subject
- 组

- 描述

- 文档参考
- 推荐操作
- 估计的复杂性

第二个 CSV 文件的名称中包含 Action_Items_Summary 后缀，并包含有关所有操作项出现次数的信息。

在以下示例中，学习曲线工作量列中的值表示设计转换每个操作项的方法所需的工作量。转换操作项出现次数的工作量列中的值表示按照设计的方法转换每个操作项所需的工作量。用于表示所需工作量的值基于加权量表，范围从低（最小）到高（最大）。

Schema	Action item	Number of occurrences	Learning curve efforts	Efforts to convert an occurrence of the action item
TEST.dbo	609	1	8	0.3
TEST.dbo	681	2	0.1	0.1
TEST.dbo	690	1	40	40
TEST.dbo	794	1	0	0.01
TEST.dbo	811	12	40	8
TEST.dbo	826	1	0	0.1
TEST.dbo	844	8	8	0.5
TEST.dbo	9997	3	0	0.3

第三个 CSV 文件的名称中包含 Summary，并包含以下摘要：

- 类别

- 对象数量
- 已自动转换的对象
- 需要简单操作的对象
- 需要中等复杂操作的对象
- 需要复杂操作的对象
- 代码总行数

。

创建数据库迁移多服务器评估报告

要确定整体环境的最佳目标方向，请创建多服务器评估报告。

多服务器评估报告根据您为要评估的每个架构定义提供的输入来评估多台服务器。架构定义包含数据库服务器连接参数和每个架构的全名。评估每个架构后，AWS SCT 生成一份多台服务器数据库迁移汇总摘要评估报告。该报告显示了每个可能的迁移目标的估计复杂性。

您可以使用 AWS SCT 为以下源数据库和目标数据库创建多服务器评估报告。

源数据库	目标数据库
Amazon Redshift	Amazon Redshift
Azure SQL 数据库	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
Azure Synapse Analytics	Amazon Redshift
BigQuery	Amazon Redshift
Greenplum	Amazon Redshift
IBM Db2 for z/OS	Amazon Aurora MySQL 兼容版（Aurora MySQL）、 Amazon Aurora PostgreSQL 兼容版（Aurora PostgreSQL）、MySQL、PostgreSQL
IBM Db2 LUW	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL
Microsoft SQL Server	Aurora MySQL、Aurora PostgreSQL、Amazon Redshift、适用于 Aurora PostgreSQL 的 Babelfish、MariaDB、Microsoft SQL Server、MySQL、PostgreSQL
MySQL	Aurora PostgreSQL、MySQL、PostgreSQL
Netezza	Amazon Redshift
Oracle	Aurora MySQL、Aurora PostgreSQL、Amazon Redshift、MariaDB、MySQL、Oracle、PostgreSQL

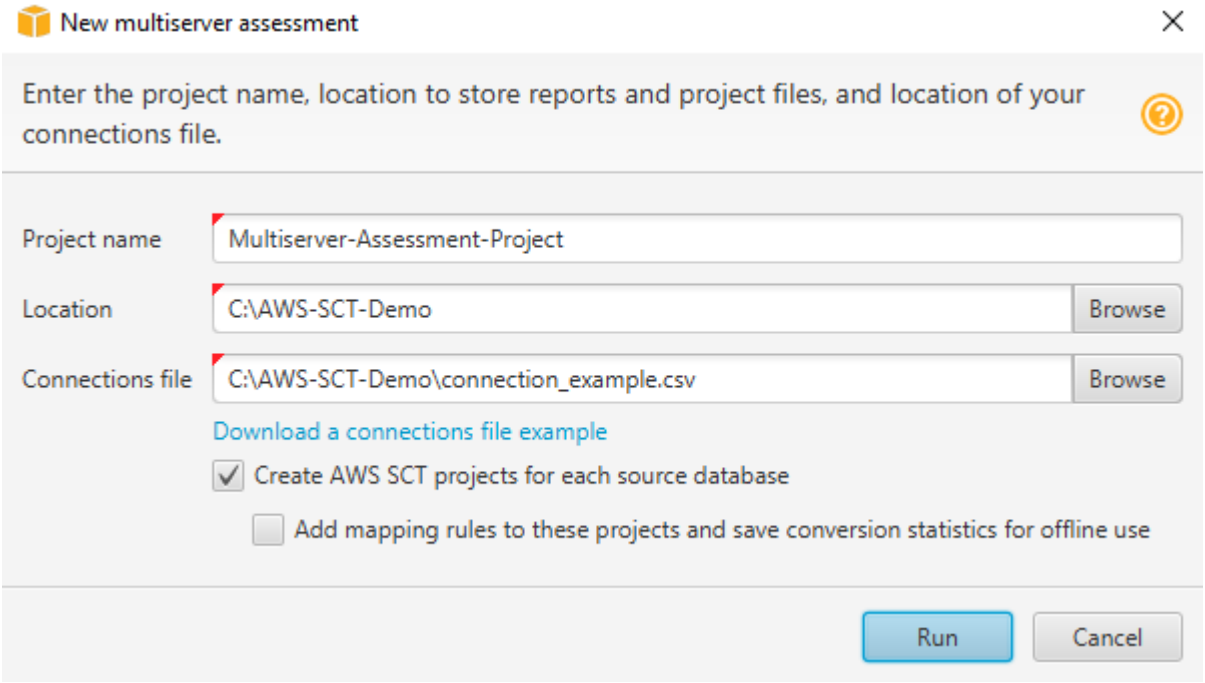
源数据库	目标数据库
PostgreSQL	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
SAP ASE	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL
Snowflake	Amazon Redshift
Teradata	Amazon Redshift
Vertica	Amazon Redshift

执行多服务器评估

使用以下过程用 AWS SCT 执行多服务器评估。您无需在 AWS SCT 中创建新项目便能执行多服务器评估。在开始之前，请确保您已准备好包含数据库连接参数的逗号分隔值（CSV）文件。此外，请确保已安装所有必需的数据库驱动程序，并在 AWS SCT 设置中设置驱动程序的位置。有关更多信息，请参阅下载所需的数据库驱动程序。

执行多服务器评估并创建汇总摘要报告

1. 在 AWS SCT 中，选择文件、新建多服务器评估。将打开新建多服务器评估对话框。



2. 选择下载连接文件示例，下载带有数据库连接参数的 CSV 文件的空模板。
3. 输入项目名称、位置（用于存储报告）和连接文件（CSV 文件）的值。
4. 选择为每个源数据库创建 AWS SCT 项目，以便在生成评估报告后自动创建迁移项目。
5. 打开为每个源数据库创建 AWS SCT 项目后，您可以选择向这些项目添加映射规则并保存转化统计信息以供离线使用。在这种情况下，AWS SCT 将向每个项目添加映射规则，并将源数据库元数据保存在项目中。有关更多信息，请参阅在离线模式下运行 AWS SCT。
6. 选择运行。

   此时将出现一个进度条，指示数据库评估的进度。目标引擎的数量可能会影响评估运行时系统。
7. 如果显示以下消息，请选择是：对所有数据库服务器进行全面分析可能需要一些时间。你要继续吗？

   多服务器评估报告完成后，会出现一个显示完成的屏幕。
8. 选择打开报告以查看汇总摘要评估报告。

默认情况下，AWS SCT 会为所有源数据库生成汇总报告，为源数据库中的每个架构名称生成详细的评估报告。有关更多信息，请参阅查找和查看报告。

启用 为每个源数据库创建 AWS SCT 项目 选项后，AWS SCT 将为每个源数据库创建一个空项目。AWS SCT 还会如前所述创建评估报告。在分析了这些评估报告并为每个源数据库选择迁移目标之后，将目标数据库添加到这些空项目中。

启用向这些项目添加映射规则并保存转化统计信息以供离线使用选项，AWS SCT 将为每个源数据库创建一个项目。这些项目包括以下信息：

- 源数据库和虚拟目标数据库平台。有关更多信息，请参阅使用虚拟目标。
- 此源-目标对的映射规则。有关更多信息，请参阅创建映射规则。
- 此源-目标对的数据库迁移评估报告。
- 源架构元数据，使您能够在离线模式下使用此 AWS SCT 项目。有关更多信息，请参阅在离线模式下运行 AWS SCT。

准备输入 CSV 文件

要提供连接参数作为多服务器评估报告的输入，请使用 CSV 文件，如以下示例中所示。

```
Name,Description,Secret Manager Key,Server IP,Port,Service Name,Database name,BigQuery
 path,Source Engine,Schema Names,Use Windows Authentication,Login,Password,Use
 SSL,Trust store,Key store,SSL authentication,Target Engines
Sales,,,192.0.2.0,1521,pdb,,,ORACLE,Q4_2021;FY_2021,,user,password,,,,,POSTGRESQL;AURORA_POSTGR
Marketing,,,ec2-a-b-c-d.eu-
west-1.compute.amazonaws.com,1433,,target_audience,,MSSQL,customers.dbo,,user,password,,,,,AURO
HR,,,192.0.2.0,1433,,employees,,MSSQL,employees.%,true,,,,,,,AURORA_POSTGRESQL
Customers,,secret-name,,,,,,MYSQL,customers,,,,,,,,AURORA_POSTGRESQL
Analytics,,,198.51.100.0,8195,,STATISTICS,,DB2LUW,BI_REPORTS,,user,password,,,,,POSTGRESQL
Products,,,203.0.113.0,8194,,,,TERADATA,new_products,,user,password,,,,,REDSHIFT
```

前面的示例使用分号分隔 Sales 数据库的两个架构名称。它还使用分号来分隔 Sales 数据库的两个目标数据库迁移平台。

此外，前面的示例使用 AWS Secrets Manager 连接到 Customers 数据库并使用 Windows 身份验证来连接到 HR 数据库。

您可以创建新的 CSV 文件或从 AWS SCT 中下载 CSV 文件的模板并填写所需信息。确保 CSV 文件的第一行包含与前面示例中所示相同的列名。

下载输入 CSV 文件的模板

1. 启动 AWS SCT。
2. 选择文件，然后选择新建多服务器评估。
3. 选择下载连接文件示例。

确保您的 CSV 文件包含模板提供的以下值：

- 名称：帮助识别数据库的文本标签。AWS SCT 在评估报告中显示此文本标签。
- 描述：一个可选值，您可以在其中提供有关数据库的其他信息。
- Secret Manager 密钥：将您的数据库凭证存储在 AWS Secrets Manager 中的密钥的名称。要使用 Secrets Manager，请确保将 AWS 配置文件存储在中 AWS SCT。有关更多信息，请参阅使用 AWS Secrets Manager。

> **Important**
>
> 如果输入文件中包含服务器 IP、端口、登录名和密码参数，则 AWS SCT 会忽略Secret Manager 密钥参数。

- 服务器 IP：源数据库服务器的域名服务（DNS）名称或 IP 地址。
- 端口：用于连接到源数据库服务器的端口。
- 服务名称：如果您使用服务名连接您的 Oracle 数据库，则为要连接的 Oracle 服务的名称。
- 数据库名称：数据库的名称。对于 Oracle 数据库，使用 Oracle 系统 ID（SID）。
- BigQuery 路径：源 BigQuery 数据库的服务帐号密钥文件的路径。有关创建此文件的更多信息，请参阅 作为源的 BigQuery 的权限。
- 源引擎：源数据库的类型。使用下列值之一：
  - AZURE_MSSQL：适用于 Azure SQL 数据库。
  - AZURE_SYNAPSE：适用于 Azure Synapse Analytics 数据库。
  - GOOGLE_BIGQUERY：适用于 BigQuery 数据库。
  - DB2ZOS：适用于 IBM Db2 for z/OS 数据库。
  - DB2LUW：适用于 IBM Db2 LUW 数据库。
  - GREENPLUM：适用于 Greenplum 数据库。
  - MSSQL：适用于 Microsoft SQL Server 数据库。
  - MYSQL：适用于 MySQL 数据库。
  - NETEZZA：适用于 Netezza 数据库。
  - ORACLE：适用于 Oracle 数据库。
  - POSTGRESQL：适用于 PostgreSQL 数据库。
  - REDSHIFT：适用于 Amazon Redshift 数据库
  - SNOWFLAKE：适用于 Snowflake 数据库。

  - SYBASE_ASE：适用于 SAP ASE 数据库。
  - TERADATA：适用于 Teradata 数据库。
  - VERTICA：适用于 Vertica 数据库。
- 架构名称：要包含在评估报告中的数据库架构的名称。

  对于 Azure SQL 数据库、Azure Synapse Analytics、BigQuery、Netezza、SAP ASE、Snowflake 和 SQL Server，请使用以下格式的架构名称：

  db_name.*schema_name*

  将 *db_name* 替换为源数据库的名称。

  将 *schema_name* 替换为源架构的名称。

  用双引号将包含点的数据库或架构名称括起来，如下所示:`"database.name"."schema.name"`。

  使用分号分隔多个架构名称，如下所示:`Schema1;Schema2`。

  数据库和架构名称区分大小写。

  使用百分比 (%) 作为通配符来替换数据库或架构名称中任意数量的任何符号。前面的示例使用百分比 (%) 作为通配符，将 `employees` 数据库中的所有架构包含在评估报告中。
- 使用 Windows 身份验证：如果你使用 Windows 身份验证连接到 Microsoft SQL Server 数据库，请输入 true。有关更多信息，请参阅当将 Microsoft SQL Server 用作源时使用 Windows 身份验证。
- 登录名：用于连接到源数据库服务器的用户名。
- 密码：用于连接到源数据库服务器的密码。
- 使用 SSL：如果您使用安全套接字层 (SSL) 连接到源数据库，请输入 true。
- 信任存储 ：用于 SSL 连接的信任存储。
- 密钥存储：用于 SSL 连接的密钥存储。
- SSL 身份验证：如果您使用通过证书进行的 SSL 身份验证，请输入 true。
- 目标引擎：目标数据库平台。使用以下值在评估报告中指定一个或多个目标：
  - AURORA_MYSQL：适用于 Aurora MySQL 兼容数据库。
  - AURORA_POSTGRESQL：适用于 Aurora PostgreSQL 兼容数据库
  - BABELFISH：适用于 Aurora PostgreSQL 的 Babelfish 数据库。
  - MARIA_DB：适用于 MariaDB 数据库。
  - MSSQL：适用于 Microsoft SQL Server 数据库。

- MYSQL：适用于 MySQL 数据库。
- ORACLE：适用于 Oracle 数据库。
- POSTGRESQL：适用于 PostgreSQL 数据库。
- REDSHIFT：适用于 Amazon Redshift 数据库。

使用分号分隔多个目标，如下所示：`MYSQL;MARIA_DB`。目标的数量会影响运行评估所需的时间。

查找和查看报告

多服务器评估生成两种类型的报告：

- 所有源数据库的汇总报告。
- 源数据库中每个架构名称的目标数据库的详细评估报告。

报告存储在您在新建多服务器评估对话框中为位置选择的目录中。

要访问详细报告，您可以浏览子目录，这些子目录按源数据库、架构名称和目标数据库引擎组织。

汇总报告分四列显示有关目标数据库转换复杂性的信息。这些列包含有关代码对象、存储对象、语法元素和转换复杂性的转换的信息。

以下示例显示了将两个 Oracle 数据库架构转换为适用于 PostgreSQL 的 Amazon RDS 的信息。

Server IP address and port	Secret Manager key	Name	Description	Database name	Schema name	Code object conversion % for "Amazon RDS for PostgreSQL"	Storage object conversion % for "Amazon RDS for PostgreSQL"	Syntax Elements conversion % for "Amazon RDS for PostgreSQL"	Conversion Complexity for "Amazon RDS for PostgreSQL"
192.0.2.0:1521		Sales		ORCL	Q4_2021	97.78%	100.00%	98.76%	1
192.0.2.0:1521		Sales		pdb	FY_2021	82.35%	85.19%	99.24%	10

对于每个指定的其他目标数据库引擎，都会在报告中追加相同的四列。

有关如何阅读这些信息的详细信息，请参阅以下内容。

汇总评估报告的输出

AWS Schema Conversion Tool 中的多服务器数据库迁移汇总评估报告是一个 CSV 文件，其中包含以下几列：

- `Server IP address and port`
- `Secret Manager key`

- Name
- Description
- Database name
- Schema name
- Code object conversion % for *target_database*
- Storage object conversion % for *target_database*
- Syntax elements conversion % for *target_database*
- Conversion complexity for *target_database*

要收集信息，AWS SCT 运行完整的评估报告，然后按架构汇总报告。

在报告中，以下三个字段显示了基于评估可能自动转换的百分比：

代码对象转化率%

AWS SCT 可以自动转换或只需最少更改即可转换架构中代码对象的百分比。代码对象包括过程、函数、视图等。

存储对象转换率%

SCT 可以自动转换或只需最少更改即可转换存储对象的百分比。存储对象包括表、索引、约束等。

语法元素转换率%

SCT 可以自动转换的语法元素的百分比。语法元素包括 SELECT、FROM、DELETE 和 JOIN 子句等。

转换复杂度计算基于操作项的概念。操作项反映了源代码中发现的一种问题，在迁移到特定目标的过程中，您需要手动修复这些问题。一个操作项可以多次出现。

加权量表确定了执行迁移的复杂程度。数字 1 代表最低的复杂度，数字 10 代表最高的复杂度。

使用 AWS SCT 转换数据库架构

您可以使用 AWS Schema Conversion Tool（AWS SCT）将现有的数据库架构从一个数据库引擎转换为另一个数据库引擎。使用 AWS SCT 用户界面转换数据库相当简单，但在进行转换之前还有一些方面需要考虑。

例如，您可以使用 AWS SCT 执行以下操作：

- 您可以使用 AWS SCT 将现有本地数据库架构复制到运行相同引擎的 Amazon RDS 数据库实例。您可以使用此功能来分析迁移到云和更改许可证类型的潜在成本节省。
- 在某些情况下，数据库功能无法转换为等效的 Amazon RDS 功能。如果您在 Amazon Elastic Compute Cloud (Amazon EC2) 平台上托管并自行管理数据库，则可通过替代 AWS 服务模拟这些功能。
- AWS SCT 可自动执行将联机事务处理（OLTP）数据库架构转换为 Amazon Relational Database Service（Amazon RDS）MySQL 数据库实例、Amazon Aurora 数据库集群或 PostgreSQL 数据库实例的大部分过程。源数据库和目标数据库引擎包含许多不同的特性和功能，AWS SCT 尽可能尝试在您的 Amazon RDS 数据库实例中创建等效架构。如果无法直接转换，AWS SCT 会提供一个列表，其中包含可供您采取的操作。

主题

AWS SCT 支持以下联机事务处理（OLTP）转换。

源数据库	目标数据库
IBM Db2 for z/OS（版本 12）	Amazon Aurora MySQL 兼容版本、Amazon Aurora PostgreSQL 兼容版本、MySQL、PostgreSQL
IBM Db2 LUW（版本 9.1、9.5、9.7、10.5、11.1 和 11.5）	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL
Microsoft Azure SQL 数据库	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
Microsoft SQL Server（版本 2008 R2 及更高版本）	Aurora MySQL、Aurora PostgreSQL、适用于 Aurora PostgreSQL 的 Babelfish、MariaDB、Microsoft SQL Server、MySQL、PostgreSQL
MySQL（版本 5.5 及更高版本）	Aurora PostgreSQL、MySQL、PostgreSQL 您可以将架构和数据从 MySQL 迁移到 Aurora MySQL 数据库集群，而无需使用 AWS SCT。有关更多信息，请参阅将数据迁移到 Amazon Aurora 数据库集群。
Oracle（版本 10.2 及更高版本）	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、Oracle、PostgreSQL
PostgreSQL（版本 9.1 及更高版本）	Aurora MySQL、Aurora PostgreSQL、MySQL、PostgreSQL
SAP ASE（12.5、15.0、15.5、15.7 和 16.0）	Aurora MySQL、Aurora PostgreSQL、MariaDB、MySQL、PostgreSQL

有关转换数据仓库架构的更多信息，请参阅 使用 AWS SCT 将数据仓库架构转换为 Amazon Redshift。

要将数据库架构转换为 Amazon RDS，请执行以下概括步骤：

- 在 AWS SCT 中创建迁移规则：使用 AWS SCT 转换架构之前，您可以设置以下操作的规则：更改列数据类型、将对象从一个架构复制到另一架构，以及更改对象名称。

- 使用 AWS SCT 转换架构：AWS SCT 创建本地版本的转换后架构以供查看，但在您做好准备之前，不会将其应用于目标数据库实例。
- 使用 AWS SCT 创建评估报告：AWS SCT 创建数据库迁移评估报告，详细介绍无法自动转换的架构元素。您可以使用此报告来确定需要在与源数据库兼容的 Amazon RDS 数据库实例中的哪个位置创建架构。
- 在 AWS SCT 中处理手动转换：如果存在无法自动转换的架构元素，您有两种选择：更新源架构，然后再次转换；或者在目标 Amazon RDS 数据库实例中创建等效的架构元素。
- 更新和刷新 AWS SCT 中的转换后的架构：您可以使用源数据库中的最新架构更新 AWS SCT 项目。
- 保存和应用 AWS SCT 中的转换后架构：在您准备就绪后，可让 AWS SCT 将本地项目中的转换后的架构应用于目标 Amazon RDS 数据库实例。

在 AWS SCT 中创建迁移规则

在使用 AWS SCT 转换架构之前，您可以设置迁移规则。您可以在 AWS SCT 中设置以下操作的迁移规则：更改列数据类型、将对象从一个架构移动到另一架构以及更改对象名称。例如，假定您的源架构中有一组名为 `test_TABLE_NAME` 的表。您可以设置一条规则，将前缀 `test_` 更改为目标架构中的前缀 `demo_`。

> **Note**
>
> 您只能为不同的源数据库引擎和目标数据库引擎创建迁移规则。

您可以创建执行以下任务的迁移规则：

- 添加、删除或替换前缀
- 添加、删除或替换后缀
- 更改列排序规则
- 更改数据类型
- 更改 `char`、`varchar`、`nvarchar` 和 `string` 数据类型的长度
- 移动对象
- 重命名对象

您可以为以下对象创建迁移规则：

- 数据库
- 架构
- 表
- 列

创建迁移规则

您可以创建迁移规则并将规则另存为项目的一部分。打开项目，使用以下过程创建迁移规则。

创建迁移规则

1. 在视图菜单上，选择映射视图。
2. 在服务器映射中，选择一对源服务器和目标服务器。
3. 选择新建迁移规则。此时显示转换规则对话框。
4. 选择 Add new rule。规则列表中新增一行。
5. 配置规则：

    a. 对于 Name（名称），请为规则输入一个名称。

    b. 对于 For，请选择该规则适用的对象的类型。

    c. 对于 where，请输入在应用迁移规则之前要应用于对象的筛选器。通过使用 LIKE 子句对 WHERE 子句进行评估。您可以输入一个确切名称以选择一个对象，也可以输入一种模式来选择多个对象。

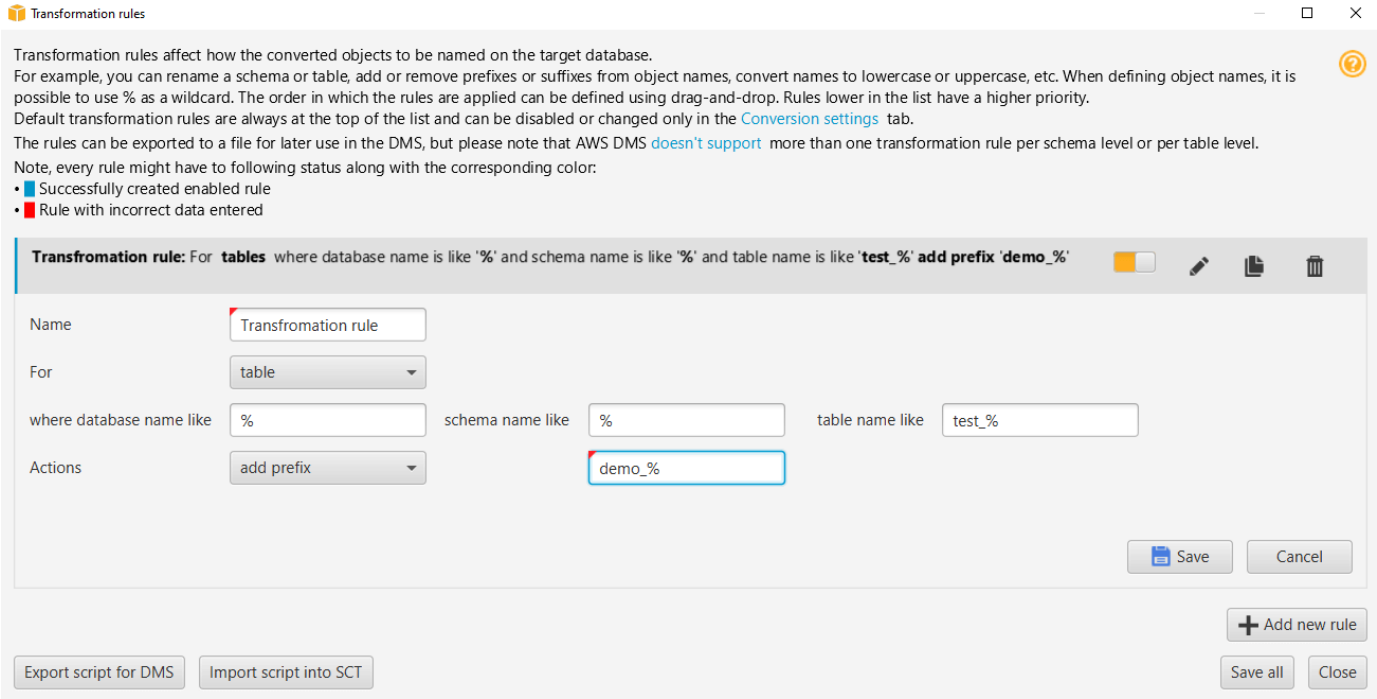
    适用于 WHERE 子句的字段有所不同，具体取决于对象类型。例如，如果对象类型为架构，则只有一个字段可用于架构名称。

    d. 对于操作，选择要创建的迁移规则的类型。

    e. 根据规则类型，输入一个或两个其他值。例如，要重命名对象，请输入对象的新名称。要替换前缀，请输入旧前缀和新前缀。

    对于 char、varchar、nvarchar 和字符串数据类型，您可以使用乘法运算符更改数据类型长度。例如，`%*4` 值会将 `varchar(10)` 数据类型转换为 `varchar(40)`。

6. 配置迁移规则后，请选择保存以保存您的规则。您也可以选择 Cancel 取消所做更改。



7. 添加、编辑和删除完规则后，选择 Save All 以保存您的所有更改。
8. 选择关闭以关闭转换规则对话框。

您可以使用切换图标关闭迁移规则，而不将其删除。您可以使用复制图标复制现有的迁移规则。您可以使用铅笔图标编辑现有的迁移规则。您可以使用删除图标来删除现有的迁移规则。要保存对迁移规则所做的所有更改，请选择全部保存。

导出迁移规则

如果您使用 AWS DMS 将数据从源数据库迁移到目标数据库，则可向 AWS DMS 提供有关迁移规则的信息。有关任务的更多信息，请参阅处理 AWS Database Migration Service 复制任务。

导出迁移规则

1. 在 AWS Schema Conversion Tool 中，在视图菜单上选择映射视图。
2. 在迁移规则中，选择迁移规则，然后选择修改迁移规则。
3. 选择导出 AWS DMS 脚本。
4. 浏览到要保存脚本的位置，然后选择 Save。迁移规则另存为可由 AWS DMS 使用的 JSON 脚本。

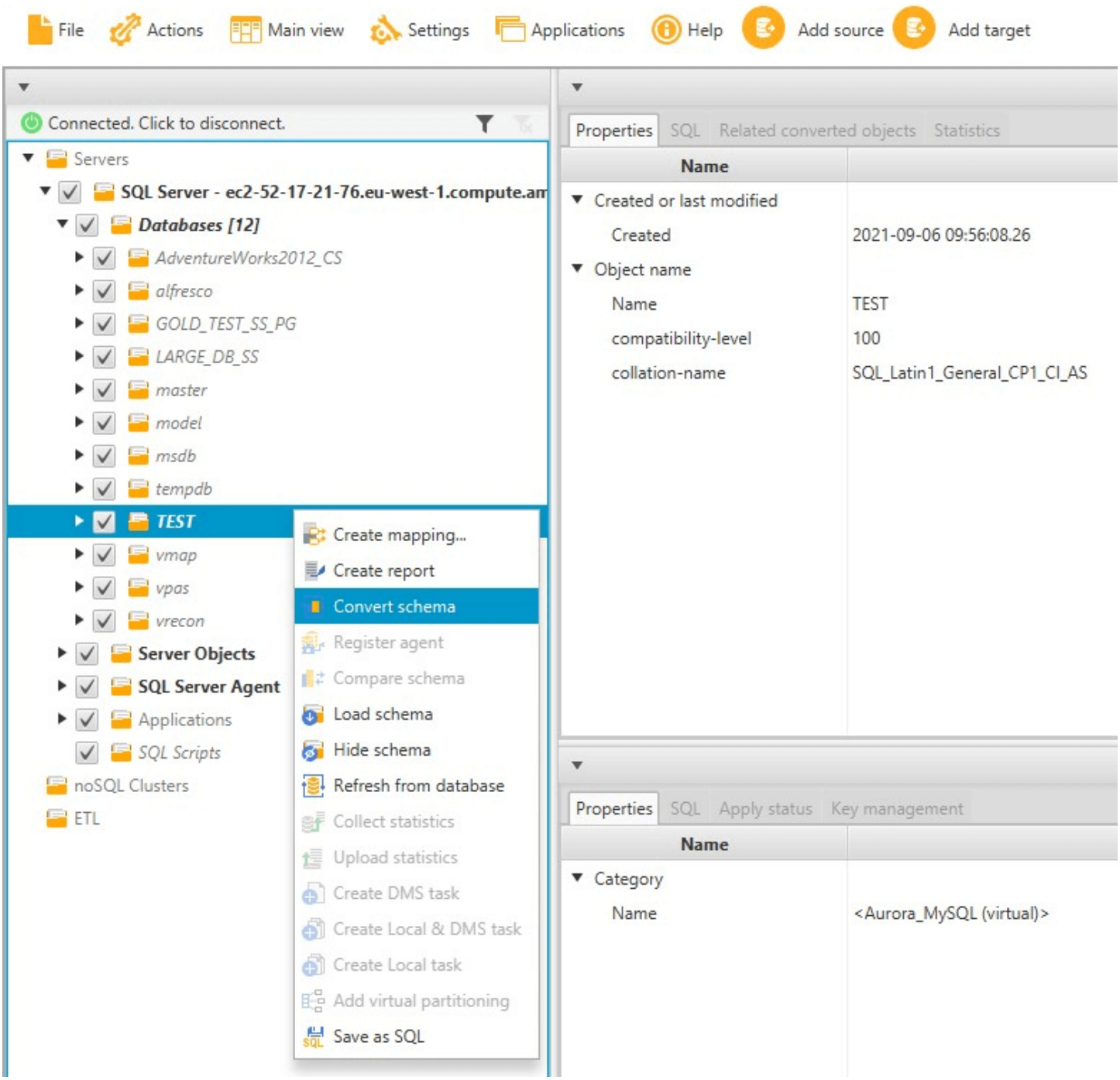
使用 AWS SCT 转换架构

将项目连接到源数据库和目标 Amazon RDS 数据库实例后，您的 AWS Schema Conversion Tool 项目会在左侧面板中显示源数据库中的架构。该架构以树状图格式显示，且树的每个节点均延迟加载。如果您选择树状图中的节点，AWS SCT 会在这个时候要求您的源数据库提供架构信息。

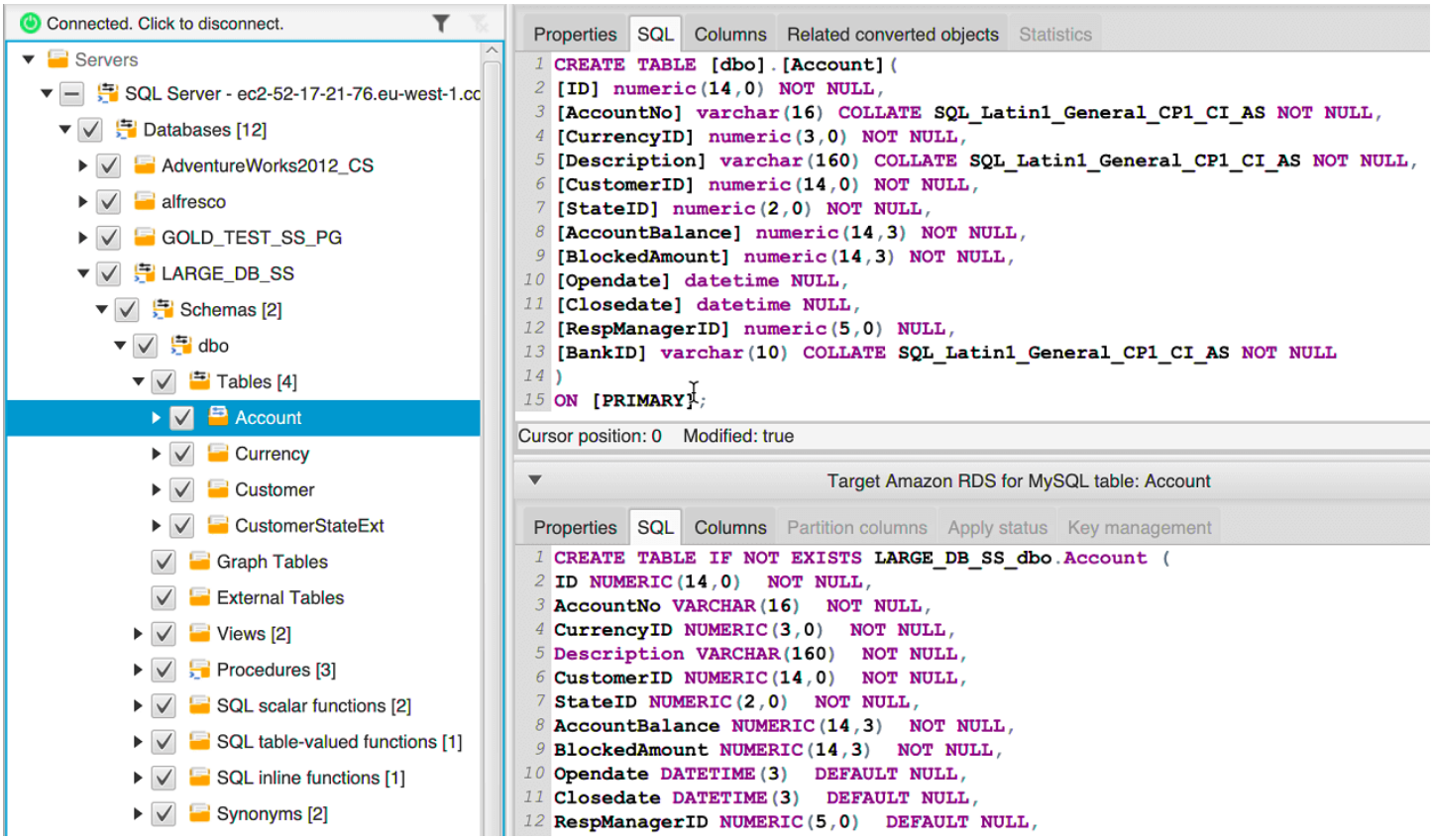
您可以从您的源数据库中选择架构项目，然后将该架构转换为目标数据库实例的数据库引擎的等效架构。您可以从源数据库中选择要转换的任何架构项目。如果您选择的架构项取决于父项，则 AWS SCT 还会为该父项生成架构。例如，假定您选择了要转换的表。如果是这样，AWS SCT 会生成表的架构以及该表所在的数据库。

转换架构

要转换源数据库中的架构，请选中要转换的架构名称对应的复选框。接下来，从项目的左侧面板中选择此架构。AWS SCT 用蓝色突出显示架构名称。打开该架构的上下文（右键单击）菜单，然后选择转换架构，如下所示。



转换完源数据库的架构后，您可以从项目左侧面板中选择架构项目，并在项目的中心面板中查看转换后的架构。中下方面板显示转换后架构的属性以及创建该架构所用的 SQL 命令，如下所示。



转换完架构后，您可以保存您的项目。源数据库中的架构信息随您的项目一起保存。此功能意味着您无需连接到源数据库即可脱机工作。如果为源数据库选择从数据库刷新，则 AWS SCT 会连接到源数据库以更新项目中的架构。有关更多信息，请参阅更新和刷新 AWS SCT 中的转换后的架构。

您可以为无法自动转换的项目创建一个数据库迁移评估报告。该评估报告对于识别和解析无法自动转换的架构项目很有用。有关更多信息，请参阅使用 AWS SCT 创建评估报告。

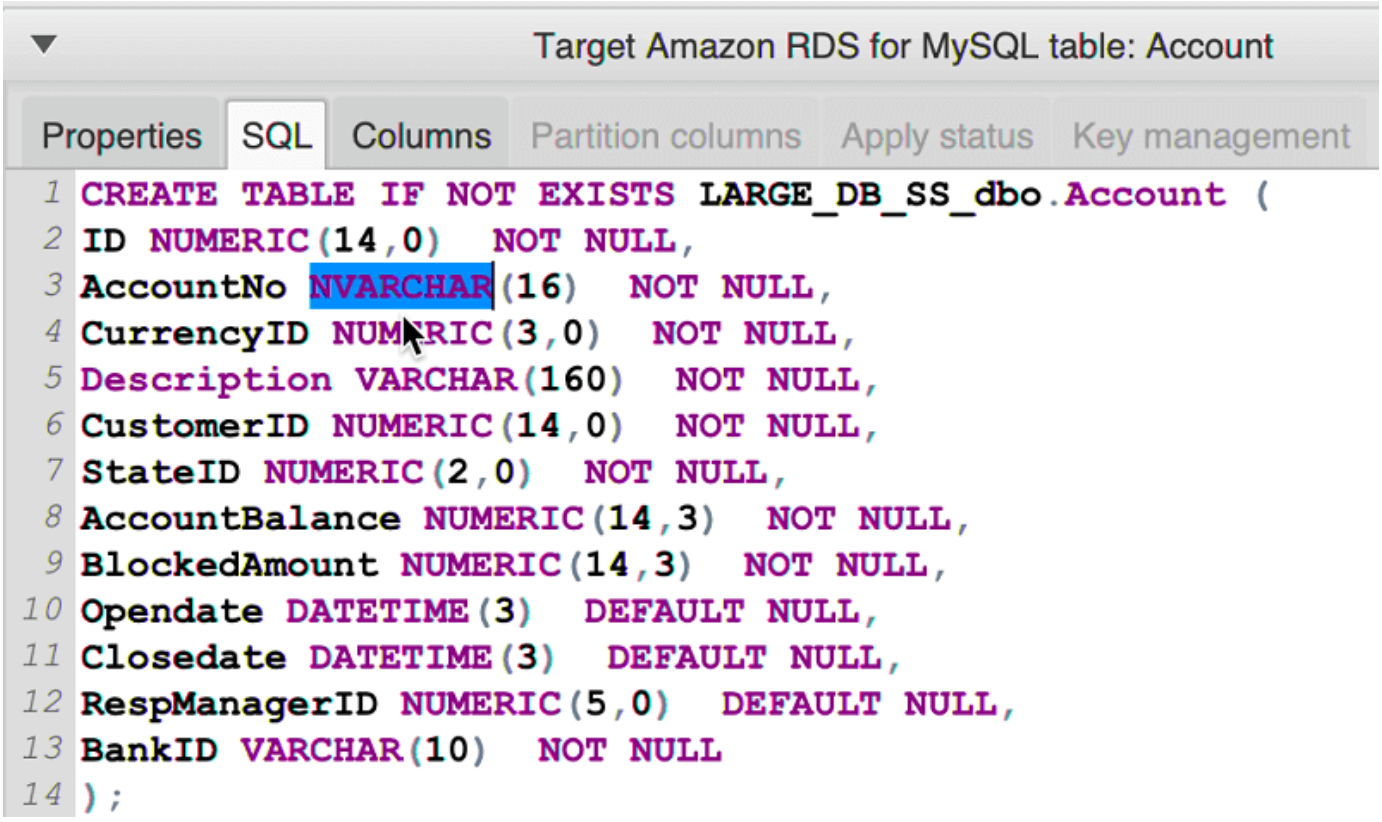
当 AWS SCT 生成转换后的架构时，不会立即将其应用于目标数据库实例。而是会将转换后的架构存储在本地，直到您准备好将其应用于目标数据库实例。有关更多信息，请参阅应用转换后的架构。

编辑转换后的架构

您可以编辑转换后的架构，并将更改另存为项目的一部分。

编辑转换后的架构

1. 在显示源数据库架构的左侧面板中，选择要为其编辑转换后架构的架构项目。
2. 在显示所选项目的转换后架构的中下方面板中，选择 SQL 选项卡。
3. 在 SQL 选项卡显示的文本中，根据需要更改架构。该架构会在您进行更新时自动随项目一起保存。



您对转换后架构的更改会在您进行更新时随项目一起存储。如果您刚从源数据库转换一个架构项目，并且已对该项目之前转换的架构进行了更新，则这些现有更新将替换为基于源数据库的新转换的架构项目。

清除转换后的架构

在您将架构应用于目标数据库实例之前，AWS SCT 将转换后的架构仅存储在本地的项目中。您可以通过选择目标数据库实例的树状图节点，然后选择从数据库刷新清除您项目中的计划架构。由于尚未向目标数据库实例中写入架构，从数据库刷新会删除 AWS SCT 项目中的计划架构元素，以便与源数据库实例中的现有元素相匹配。

在 AWS SCT 中处理手动转换

评估报告包含无法自动转换为目标 Amazon RDS 数据库实例的数据库引擎的项列表。对于无法转换的每一项，Action Items 选项卡上都有一个操作项。

您可以按如下方式应对评估报告中的操作项：

- 修改您的源数据库架构。

- 修改您的目标数据库架构。

修改源架构

对于某些项目，将源数据库中的数据库架构修改为可自动转换的架构可能更容易。首先验证新更改与您的应用程序架构兼容，然后更新源数据库中的架构。最后，用更新的架构信息刷新您的项目。然后，您可以转换更新后的架构，并生成新的数据库迁移评估报告。对于在源架构中更改的项目，不再显示操作项。

此过程的优势是，当您从源数据库刷新时，更新后的架构始终可用。

修改目标架构

对于某些项目，更容易的方法可能是将转换后的架构应用于目标数据库，然后手动将无法自动转换的项目的等效架构项目添加到目标数据库。您可以编写所有可通过应用该架构自动转换为目标数据库实例的架构。有关更多信息，请参阅保存和应用 AWS SCT 中的转换后架构。

写入到目标数据库实例的架构不包含无法自动转换的项目。在将架构应用于目标数据库实例后，您就可以在目标数据库实例中手动创建与源数据库中的架构等效的架构。数据库迁移评估报告中的操作项包含有关如何创建等效架构的建议。

> **Warning**
>
> 如果您在目标数据库实例中手动创建构架，请保存所做的所有手动操作的副本。如果再次将您的项目的转换后架构应用于目标数据库实例，它将覆盖您所做的手动操作。

在某些情况下，您无法在目标数据库实例中创建等效架构。您可能需要重新架构一部分应用程序和数据库，以便将该数据库引擎的可用功能用于您的目标数据库实例。在其他情况下，您可以简单地忽略无法自动转换的架构。

更新和刷新 AWS SCT 中的转换后的架构

您可以更新 AWS Schema Conversion Tool 项目中的源架构和目标架构。

- 源：如果您更新源数据库的架构，AWS SCT 将用源数据库中的最新架构代替项目中的架构。如果您已对源数据库的架构进行了更改，则可使用此功能来更新项目。

- 目标：如果您更新目标 Amazon RDS 数据库实例的架构，AWS SCT 将用目标数据库实例中的最新架构代替项目中的架构。如果您尚未将任何架构应用于目标数据库实例，AWS SCT 将从您的项目中清除转换后的架构。然后，您可以转换源数据库中的架构，以获得干净的目标数据库实例。

您可以通过选择从数据库刷新更新 AWS SCT 项目中的架构。

> **Note**
>
> 刷新架构时，AWS SCT 仅在需要时加载元数据。要完全加载数据库的所有架构，请打开该架构的上下文（右键单击）菜单，然后选择加载架构。例如，您可以使用此选项一次性加载数据库的元数据，然后脱机工作。

保存和应用 AWS SCT 中的转换后架构

当 AWS Schema Conversion Tool 生成转换后的架构（如 使用 AWS SCT 转换架构 中所示）后，不会立即将转换后的架构应用于目标数据库实例。而是会在本地将转换后的架构存储在项目中，直到您准备好将其应用于目标数据库实例。使用此功能，您可以使用无法自动转换为目标数据库引擎的架构项目。有关无法自动转换的项目的更多信息，请参阅使用 AWS SCT 创建评估报告。

您可以选择在将架构应用于目标数据库实例之前，让该工具将转换后的架构作为 SQL 脚本保存到文件中。此外，您还可以让该工具将转换后的架构直接应用于目标数据库实例。

将转换后的架构保存到文件中

您可以将转换后的架构作为 SQL 脚本保存到一个文本文件中。使用此方法，您可以将 AWS SCT 中生成的 SQL 脚本修改为该工具无法自动转换的地址项。然后，您可以在目标数据库实例上运行更新的脚本，将转换后的架构应用于目标数据库。

将转换后的架构另存为 SQL 脚本

1. 选择架构，并打开上下文 (右键单击) 菜单。
2. 选择另存为 SQL。
3. 输入文件名并选择保存。
4. 使用以下选项之一保存转换后的架构：
    - 单个文件

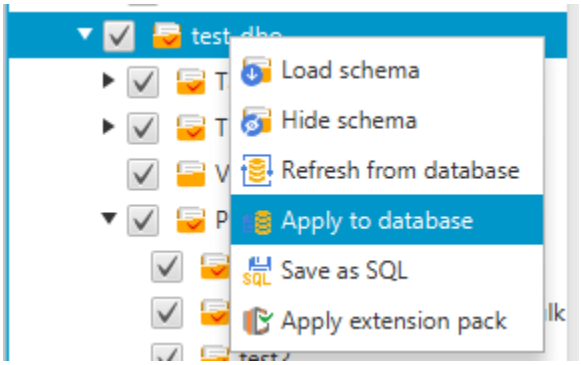
- 每个阶段单个文件
- 每个语句单个文件

选择 SQL 脚本的格式

1. 在设置菜单上，选择项目设置。
2. 选择保存脚本。
3. 对于供应商，选择数据库平台。
4. 在将 SQL 脚本保存到中，选择数据库架构脚本的保存方式。
5. 选择确定保存设置。

应用转换后的架构

在您准备好将转换后的架构应用于目标 Amazon RDS 数据库实例后，请从项目的右侧面板中选择该架构元素。打开架构元素的上下文 (右键单击) 菜单，然后选择 Apply to database，如下所示。



扩展包架构

首次将转换后的架构应用于目标数据库实例时，AWS SCT 会向目标数据库实例添加一个额外的架构。该架构用于实现将转换后的架构写入到目标数据库实例时必需的源数据库的系统功能。该架构称为扩展包架构。

不要修改扩展包架构，否则，您可能会在写入到目标数据库实例的转换后架构中遇到意外结果。将架构完全迁移到目标数据库实例后，就不再需要 AWS SCT 了，您可以删除该扩展包架构。

扩展包架构按照您的源数据库命名，如下所示：

- IBM Db2 LUW：`aws_db2_ext`

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

有关更多信息，请参阅使用 AWS SCT 扩展包中的 AWS Lambda 函数 。

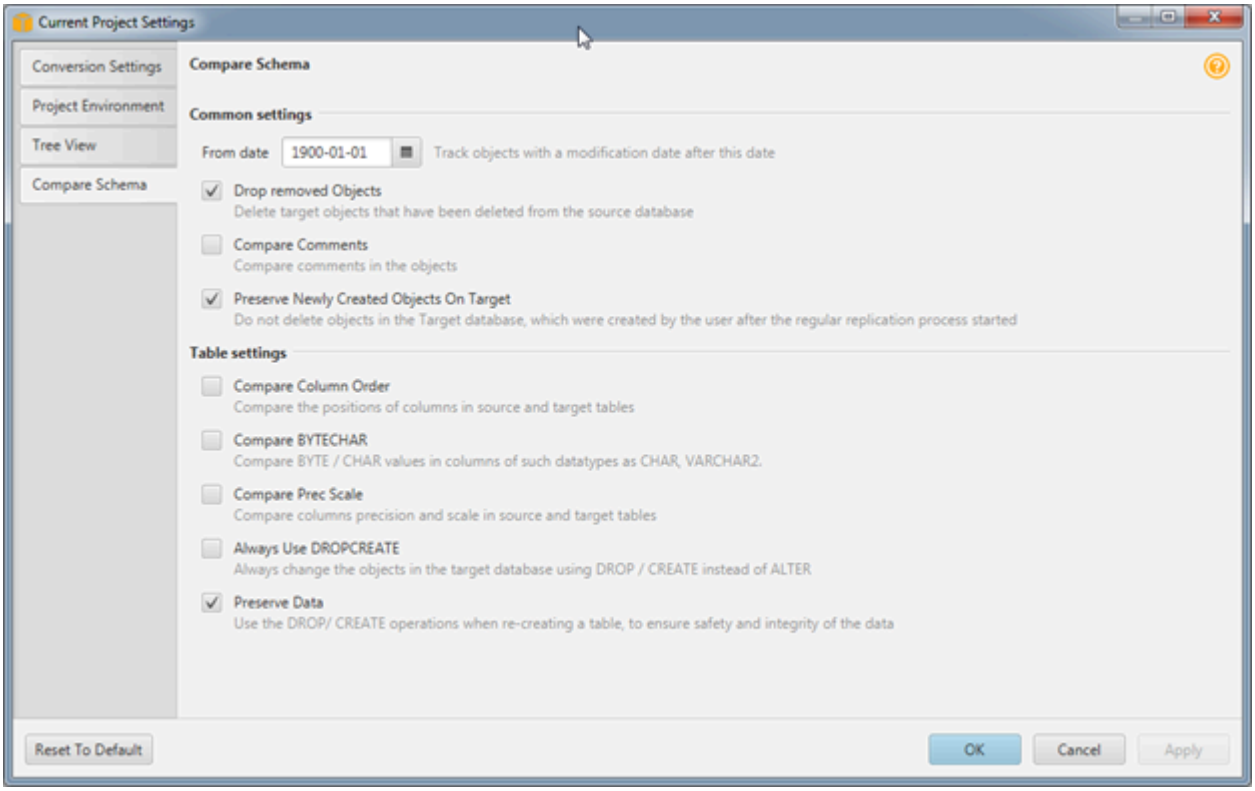
比较数据库架构

如果您在迁移之后对源或目标架构进行了更改，则可以使用 AWS SCT 对两个数据库架构进行比较。您可以对与源架构相同或更早的版本的架构进行比较。

支持以下架构比较：

- Oracle 与 Oracle，版本 12.1.0.2.0、11.1.0.7.0、11.2.0.1.0、10
- SQL Server 与 SQL Server，版本 2016、2014、2012、2008 RD2、2008
- PostgreSQL 与 PostgreSQL 以及与 Aurora PostgreSQL 兼容的版本，版本 9.6、9.5.9、9.5.4
- MySQL 与 MySQL，版本 5.6.36、5.7.17、5.5

您可以在 Project Settings (项目设置) 页的 Compare Schema (比较架构) 选项卡上指定架构比较的设置。

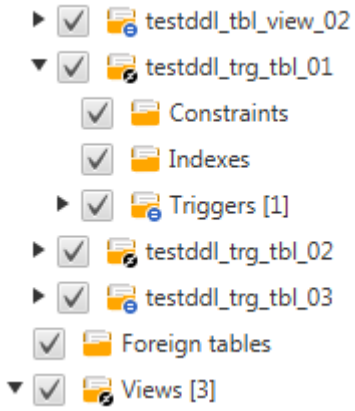


要比较架构，您可选择所需的架构，AWS SCT 会指示两个架构之间不同的对象和相同的对象。

比较两个架构

1. 打开一个现有的 AWS SCT 项目，或者创建一个项目并连接到源终端节点和目标终端节点。
2. 选择要比较的架构。
3. 打开上下文 (右键单击) 菜单并选择比较架构。

AWS SCT 将通过在对象的图标上添加黑圈来指示两个架构之间不同的对象。



您可以将架构比较结果应用于单个对象、单个对象类别或整个架构。选中要将结果应用于的类别、对象或架构旁边的框。

查找相关的已转换对象

在架构转换之后，在某些情况下 AWS SCT 可能会为源数据库上的一个架构对象创建多个对象。例如，在执行 Oracle 到 PostgreSQL 的转换时，AWS SCT 会接受每个 Oracle 触发器并在 PostgreSQL 目标上将其转换为一个触发器和一个触发器函数。此外，在 AWS SCT 将 Oracle 包函数或过程转换为 PostgreSQL 时，它会创建一个等效函数和一个 INIT 函数（应在过程或函数可以运行之前作为 init 块运行）。

以下过程可让您查看在架构转换之后创建的所有相关对象。

查看在架构转换期间创建的相关对象

1. 在架构转换之后，在目标树视图中选择已转换对象。
2. 选择 Related Converted Objects (相关的已转换对象) 选项卡。
3. 查看相关目标对象的列表。

使用 AWS SCT 将数据仓库架构转换为 Amazon Redshift

AWS Schema Conversion Tool（AWS SCT）可自动完成从数据仓库架构到 Amazon Redshift 数据库架构的大部分转换过程。因为源数据库引擎和目标数据库引擎可以具有许多不同的特性和功能，所以 AWS SCT 会尝试尽可能在您的目标数据库中创建等效架构。如果无法直接转换，AWS SCT 会提供一个评估报告，其中包含可供您采取的操作的列表。使用 AWS SCT，您可以管理密钥，映射数据类型和对象以及创建手动转换。

AWS SCT 可以将以下数据仓库架构转换为 Amazon Redshift。

- Amazon Redshift
- Azure Synapse Analytics（版本 10）
- BigQuery
- Greenplum 数据库（版本 4.3）
- Microsoft SQL Server（版本 2008 及更高版本）
- Netezza（版本 7.0.3 及更高版本）
- Oracle（版本 10.2 及更高版本）
- Snowflake（版本 3）
- Teradata（版本 13 及更高版本）
- Vertica（版本 7.2 及更高版本）

有关转换联机事务处理（OLTP）数据库架构的信息，请参阅 使用 AWS SCT 转换数据库架构。

要转换数据仓库架构，请执行以下步骤：

1. 指定优化策略和规则，并指定您希望 AWS SCT 使用的迁移规则。您可以设置以下操作的规则：更改列数据类型、将对象从一个架构复制到另一架构，以及更改对象名称。

   您可以在设置中指定优化和迁移规则。有关优化策略的更多信息，请参阅 选择 AWS SCT 所用的优化策略和规则。有关迁移规则的更多信息，请参阅 在 AWS SCT 中创建迁移规则
2. 提供有关您的源数据仓库的统计数据，以便 AWS SCT 可以优化数据仓库的转换方式。您可以直接从数据库收集统计数据，也可以上传现有的统计数据文件。有关提供数据仓库统计数据的更多信息，请参阅收集或上传 AWS SCT 的统计信息。

3. 创建数据库迁移评估报告，详细介绍无法自动转换的架构元素。您可以使用此报告来确定需要在与源数据库兼容的目标数据库中的哪个位置手动创建架构。有关评估报告的更多信息，请参阅使用 AWS SCT 创建评估报告。
4. 转换架构。AWS SCT 可以创建本地版本的转换后架构以供您查看，但在您做好准备之前，不会将其应用于目标数据库。有关转换的更多信息，请参阅使用 AWS SCT 转换架构。
5. 转换架构后，您可以管理并编辑键。键管理是数据仓库转换的核心。有关管理密钥的更多信息，请参阅在 AWS SCT 中管理和自定义键。
6. 如果存在无法自动转换的架构元素，您有两种选择：更新源架构，然后再次转换；或者在目标数据库中创建等效的架构元素。有关手动转换架构元素的更多信息，请参阅在 AWS SCT 中处理手动转换。有关更新源架构的更多信息，请参阅更新和刷新 AWS SCT 中的转换后的架构。
7. 当您准备就绪后，可以将转换后的架构应用于目标数据库。有关保存和应用转换的架构的更多信息，请参阅保存和应用 AWS SCT 中的转换后的架构。

将 Amazon Redshift 作为目标的权限

以下列出了 Amazon Redshift 作为目标所需的权限：

- CREATE ON DATABASE：允许在数据库中创建新架构。
- CREATE ON SCHEMA：允许在数据库架构中创建对象。
- GRANT USAGE ON LANGUAGE：允许在数据库中创建新的函数和过程。
- GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog：为用户提供有关 Amazon Redshift 集群的系统信息。
- GRANT SELECT ON pg_class_info：为用户提供有关表分配方式的信息。

您可以使用以下代码示例创建数据库用户并授予权限。

```
CREATE USER user_name PASSWORD your_password;
GRANT CREATE ON DATABASE db_name TO user_name;
GRANT CREATE ON SCHEMA schema_name TO user_name;
GRANT USAGE ON LANGUAGE plpythonu TO user_name;
GRANT USAGE ON LANGUAGE plpgsql TO user_name;
GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog TO user_name;
GRANT SELECT ON pg_class_info TO user_name;
GRANT SELECT ON sys_serverless_usage TO user_name;
GRANT SELECT ON pg_database_info TO user_name;
GRANT SELECT ON pg_statistic TO user_name;
```

在前面的示例中，将 *user_name* 替换为您的用户名。然后，将 *db_name* 替换为目标 Amazon Redshift 数据库的名称。接下来，将 *schema_name* 替换为您的 Amazon Redshift 架构的名称。对每个目标架构重复 GRANT CREATE ON SCHEMA 操作，您将在其中应用转换后的代码或迁移数据。最后，将 *your_password* 替换为安全密码。

您可以在目标 Amazon Redshift 数据库上应用扩展包。扩展包是一个附加模块，用于模拟将对象转换为 Amazon Redshift 时所需的源数据库函数。有关更多信息，请参阅使用 AWS SCT 扩展包。

要执行此操作，AWS SCT 需要权限才能代表您访问 Amazon S3 存储桶。要提供此权限，请使用以下策略创建一个 AWS Identity and Access Management（IAM）用户。

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

选择 AWS SCT 所用的优化策略和规则

要优化 AWS Schema Conversion Tool 转换数据仓库架构的方式，您可以选择希望该工具使用的策略和规则。转换架构并查看建议的键后，您可以调整规则或更改策略，以获得想要的结果。

选择优化策略和规则

1. 选择 Settings (设置)，然后选择 Project settings (项目设置)。随即出现 Current project settings 对话框。
2. 在左侧窗格中，选择 Optimization Strategies。优化策略将显示在右侧窗格中且已选定默认值。
3. 对于 Strategy Sector，请选择您要使用的优化策略。可从以下选项中进行选择：

   - 使用元数据，忽略统计信息：在此策略中，只将元数据的信息用于优化决策。例如，如果源表中有多个索引，则使用源数据库排序顺序，第一个索引将成为分配键。
   - 忽略元数据，使用统计信息：在此策略中，只通过统计信息做出优化决策。此策略仅适用于提供了统计数据的表和列。有关更多信息，请参阅收集或上传 AWS SCT 的统计信息。
   - 使用元数据与统计信息：在此策略中，优化决策同时使用元数据和统计数据。

4. 选择优化策略后，您可以选择要使用的规则。可从以下选项中进行选择：

   - 使用元数据选择分配键和排序键
   - 为排序规则选择事实数据表和适当的维度
   - 分析索引列的基数
   - 在查询日志表中查找最常用的表和列

   对于每个规则，您可以分别为排序键和分配键输入一个权重。AWS SCT 使用您在转换架构时选择的权重。稍后，当您查看建议的键时，如果您对结果不满意，可以返回此处更改您的设置。有关更多信息，请参阅在 AWS SCT 中管理和自定义键。

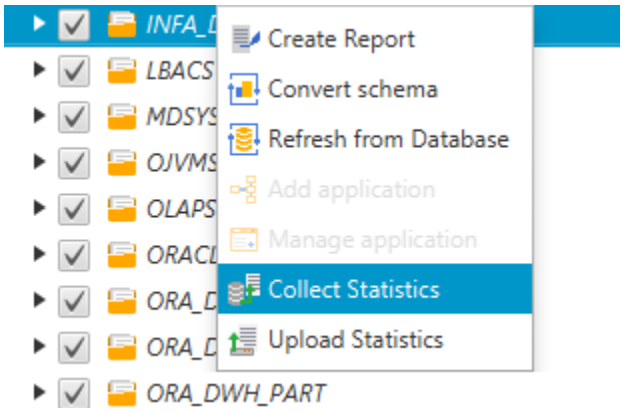
收集或上传 AWS SCT 的统计信息

要优化 AWS Schema Conversion Tool 转换数据仓库架构的方式，您可以提供该工具可使用的来自源数据库的统计数据。您可以直接从数据库收集统计数据，也可以上传现有的统计数据文件。

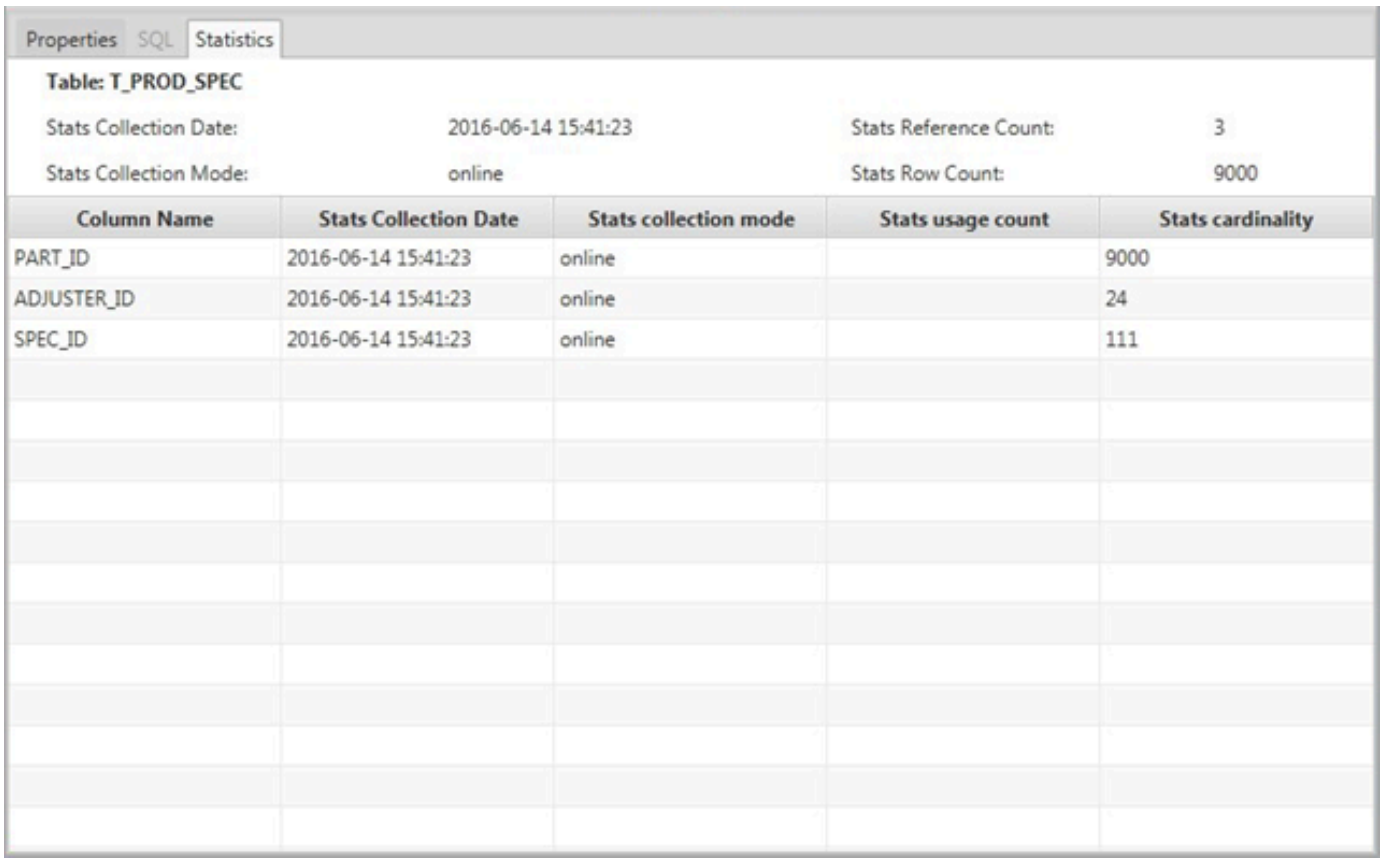
提供并查看统计数据

1. 打开您的并连接到源数据库。

2. 从项目左侧面板中选择一个架构对象，然后打开该对象的上下文 (右键单击) 菜单。选择 Collect Statistics 或 Upload Statistics，如下所示。



3. 从项目左侧面板中选择架构对象，然后选择 Statistics 选项卡。您可以查看该对象的统计数据。



稍后，当您查看建议的键时，如果您对结果不满意，可以收集更多统计数据并重复此过程。有关更多信息，请参阅在 AWS SCT 中管理和自定义键。

在 AWS SCT 中创建迁移规则

在使用 AWS SCT 转换架构之前，您可以设置迁移规则。您可以设置以下操作的迁移规则：更改列数据类型、将对象从一个架构移动到另一架构，以及更改对象名称。例如，假定您的源架构中有一组名为 `test_TABLE_NAME` 的表。您可以设置一条规则，将前缀 `test_` 更改为目标架构中的前缀 `demo_`。

> **Note**
>
> 您只能为不同的源数据库引擎和目标数据库引擎创建迁移规则。

您可以创建执行以下任务的迁移规则：

- 添加、删除或替换前缀
- 添加、删除或替换后缀
- 更改列排序规则
- 更改数据类型
- 更改 `char`、`varchar`、`nvarchar` 和 `string` 数据类型的长度
- 移动对象
- 重命名对象

您可以为以下对象创建迁移规则：

- 数据库
- 架构
- 表
- 列

创建迁移规则

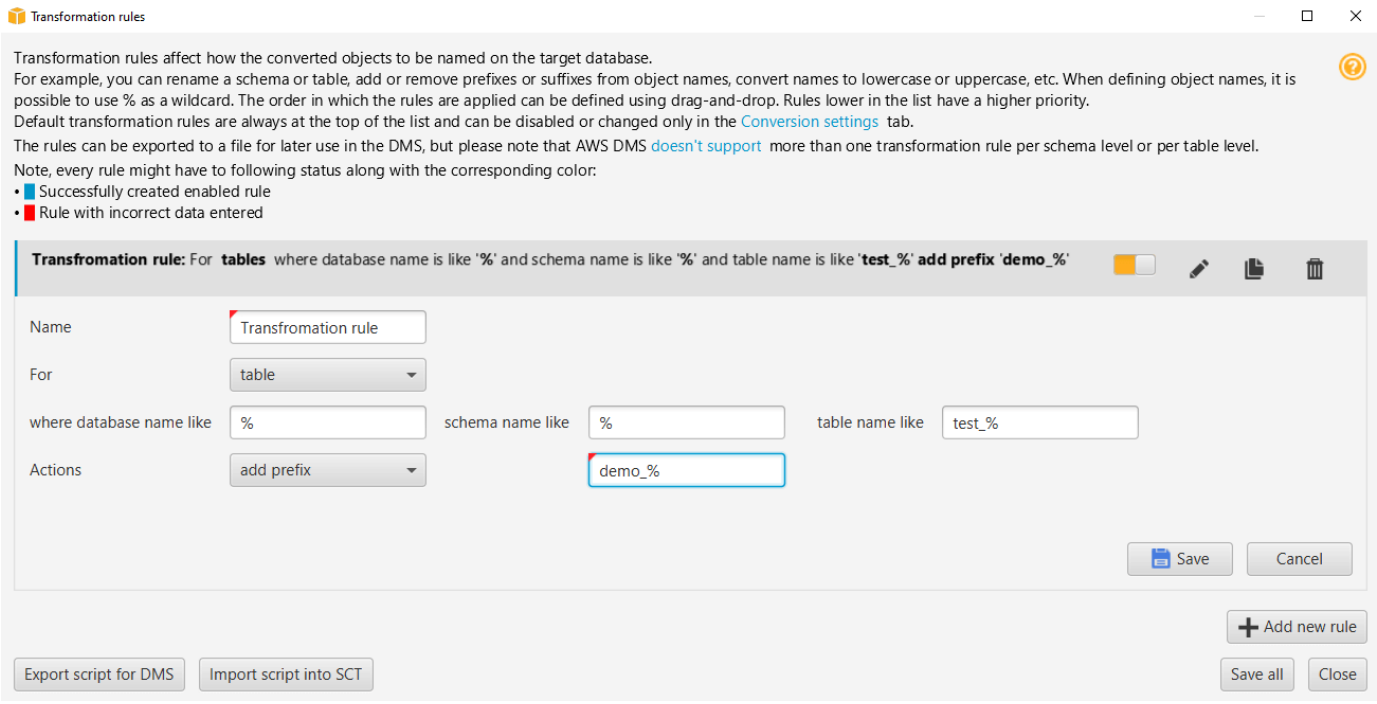
您可以创建迁移规则并将规则另存为项目的一部分。打开项目，使用以下过程创建迁移规则。

创建迁移规则

1. 在视图菜单上，选择映射视图。

2. 在服务器映射中，选择一对源服务器和目标服务器。
3. 选择新建迁移规则。此时显示转换规则对话框。
4. 选择 Add new rule。规则列表中新增一行。
5. 配置规则：
   a. 对于 Name（名称），请为规则输入一个名称。
   b. 对于 For，请选择该规则适用的对象的类型。
   c. 对于 WHERE，请输入在应用迁移规则之前要应用于对象的筛选器。通过使用 LIKE 子句对 WHERE 子句进行评估。您可以输入一个确切名称以选择一个对象，也可以输入一种模式来选择多个对象。

      适用于 WHERE 子句的字段有所不同，具体取决于对象类型。例如，如果对象类型为架构，则只有一个字段可用于架构名称。
   d. 对于操作，选择要创建的迁移规则的类型。
   e. 根据规则类型，输入一个或两个其他值。例如，要重命名对象，请输入对象的新名称。要替换前缀，请输入旧前缀和新前缀。
6. 配置迁移规则后，请选择保存以保存您的规则。您也可以选择 Cancel 取消所做更改。



7. 添加、编辑和删除完规则后，选择 Save All 以保存您的所有更改。
8. 选择关闭以关闭转换规则对话框。

您可以使用切换图标关闭迁移规则，而不将其删除。您可以使用复制图标复制现有的迁移规则。您可以使用铅笔图标编辑现有的迁移规则。您可以使用删除图标来删除现有的迁移规则。要保存对迁移规则所做的所有更改，请选择全部保存。

导出迁移规则

如果您使用 AWS Database Migration Service（AWS DMS）将数据从源数据库迁移到目标数据库，则可向 AWS DMS 提供有关迁移规则的信息。有关任务的更多信息，请参阅处理 AWS Database Migration Service 复制任务。

导出迁移规则

1. 在 AWS Schema Conversion Tool 中，在视图菜单上选择映射视图。
2. 在迁移规则中，选择迁移规则，然后选择修改迁移规则。
3. 选择导出 AWS DMS 脚本。
4. 浏览到要保存脚本的位置，然后选择 Save。迁移规则另存为可由 AWS DMS 使用的 JSON 脚本。

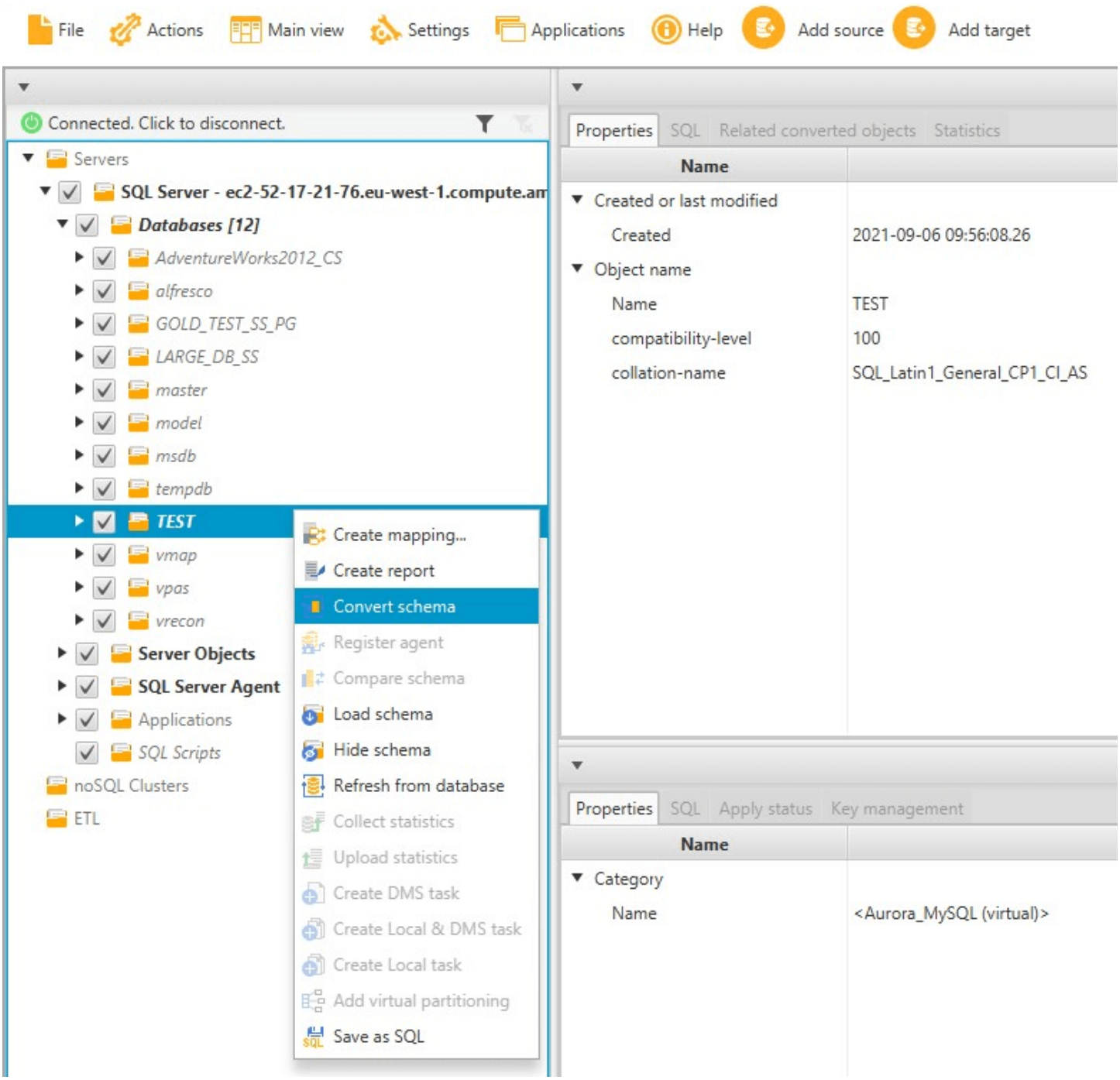
使用 AWS SCT 转换架构

将项目连接到源数据库和目标数据库后，您的 AWS Schema Conversion Tool 项目会在左侧面板中显示源数据库中的架构。该架构以树状图格式显示，且树的每个节点均延迟加载。如果您选择树状图中的节点，AWS SCT 会在这个时候要求您的源数据库提供架构信息。

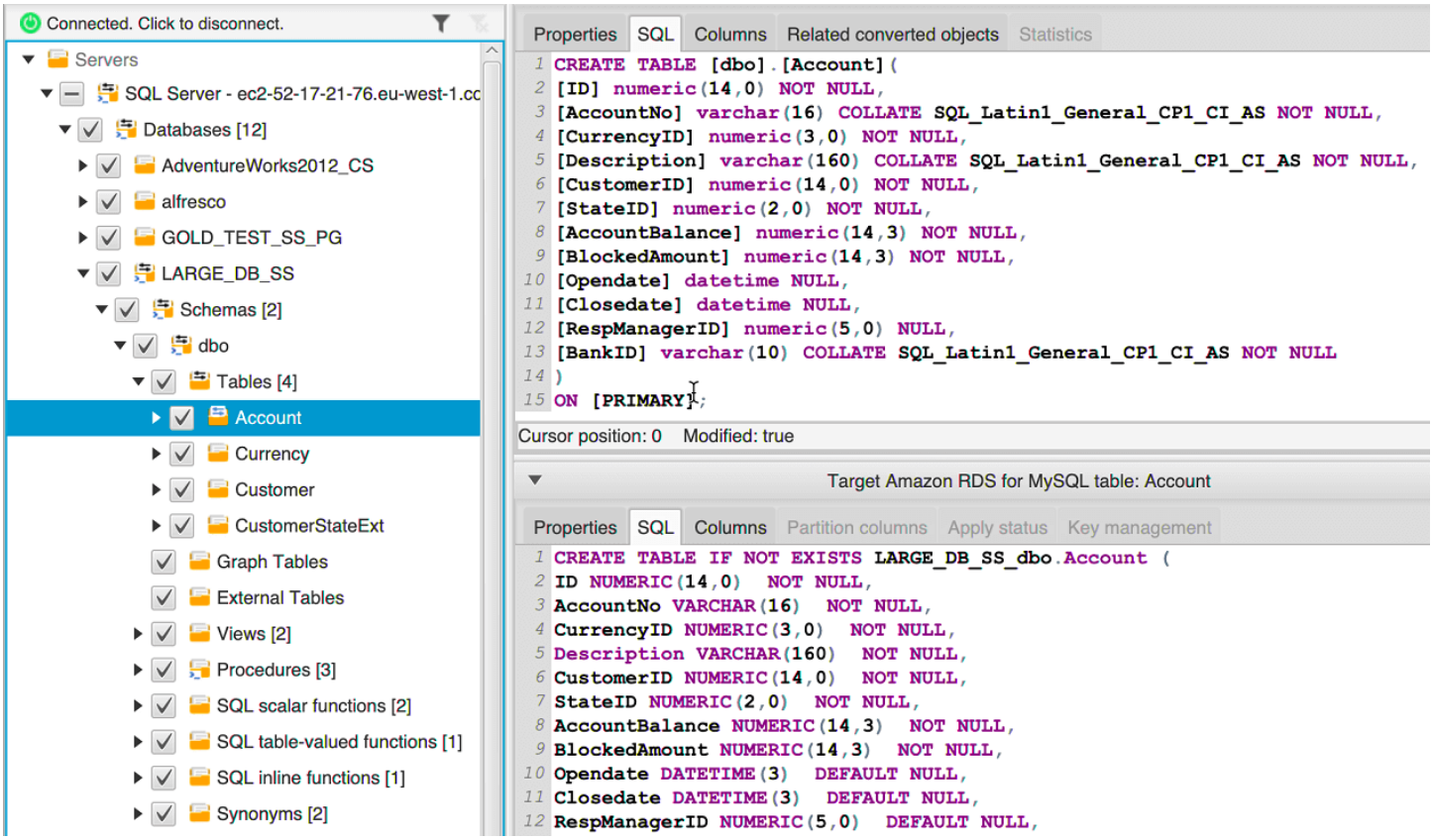
您可以从源数据库中选择架构项目，然后将该架构转换为目标数据库的数据库引擎的等效架构。您可以从源数据库中选择要转换的任何架构项目。如果您选择的架构项取决于父项，则 AWS SCT 还会为该父项生成架构。例如，如果您从表中选择一个要转换的列，则 AWS SCT 为该列、该列所在的表以及该表所在的数据库生成架构。

转换架构

要转换源数据库中的架构，请选中要转换的架构名称对应的复选框。接下来，从项目的左侧面板中选择此架构。AWS SCT 用蓝色突出显示架构名称。打开该架构的上下文 (右键单击) 菜单，然后选择转换架构，如下所示。



转换完源数据库的架构后，您可以从项目左侧面板中选择架构项目，并在项目的中心面板中查看转换后的架构。中下方面板显示转换后架构的属性以及创建该架构所用的 SQL 命令，如下所示。



转换完架构后，您可以保存您的项目。源数据库中的架构信息随您的项目一起保存。此功能意味着您无需连接到源数据库即可脱机工作。如果为源数据库选择从数据库刷新，则 AWS SCT 会连接到源数据库以更新项目中的架构。有关更多信息，请参阅更新和刷新 AWS SCT 中的转换后的架构。

您可以为无法自动转换的项目创建一个数据库迁移评估报告。该评估报告对于识别和解析无法自动转换的架构项目很有用。有关更多信息，请参阅使用 AWS SCT 创建评估报告。

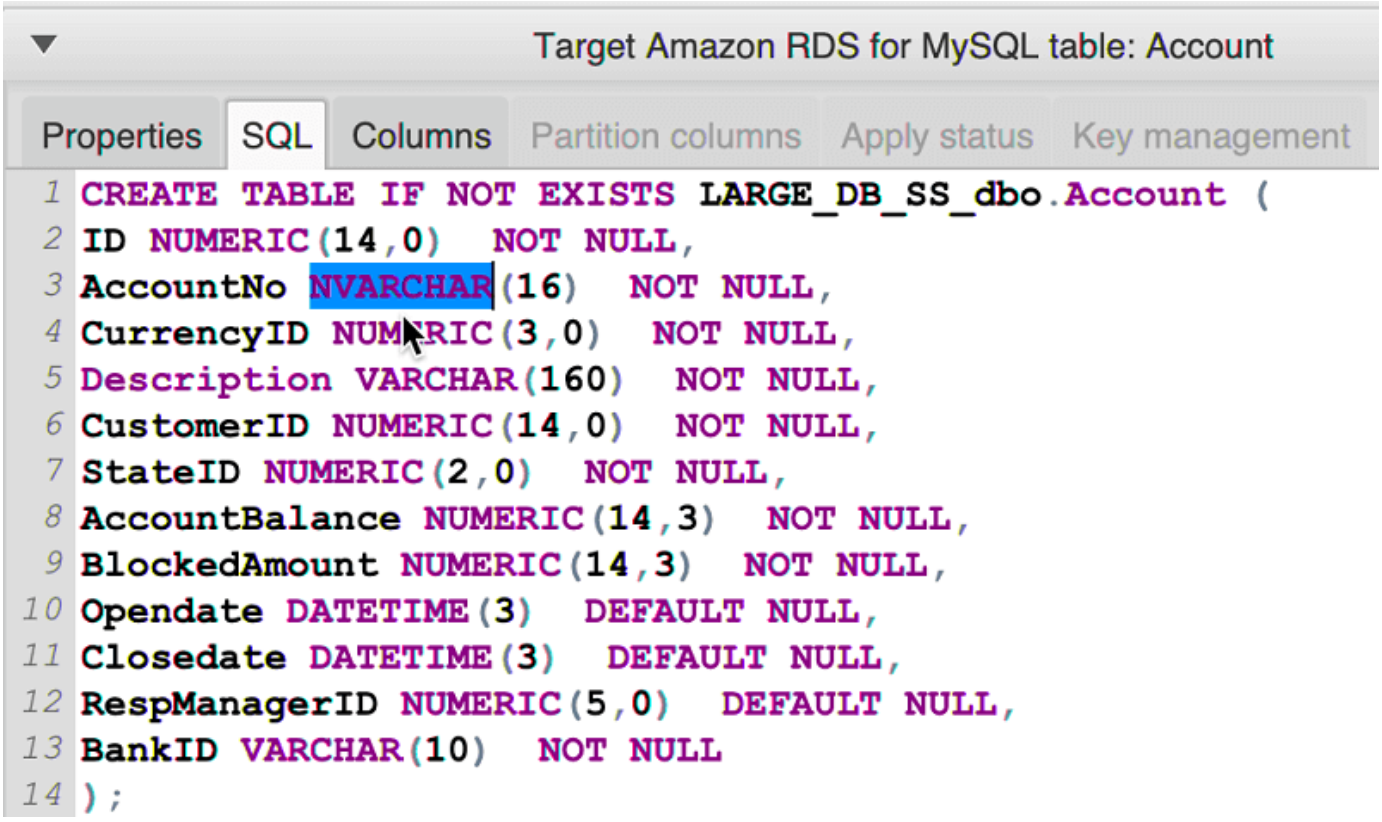
当 AWS SCT 生成转换后的架构后，不会立即将其应用于目标数据库。而会将转换后的架构存储在本地，直到您准备好将其应用到目标数据库。有关更多信息，请参阅应用转换后的架构。

编辑转换后的架构

您可以编辑转换后的架构，并将更改另存为项目的一部分。

编辑转换后的架构

1. 在显示源数据库架构的左侧面板中，选择要为其编辑转换后架构的架构项目。
2. 在显示所选项目的转换后架构的中下方面板中，选择 SQL 选项卡。
3. 在 SQL 选项卡显示的文本中，根据需要更改架构。该架构会在您进行更新时自动随项目一起保存。



您对转换后架构的更改会在您进行更新时随项目一起存储。如果您刚从源数据库转换一个架构项目，并且已对该项目之前转换的架构进行了更新，则这些现有更新将替换为基于源数据库的新转换的架构项目。

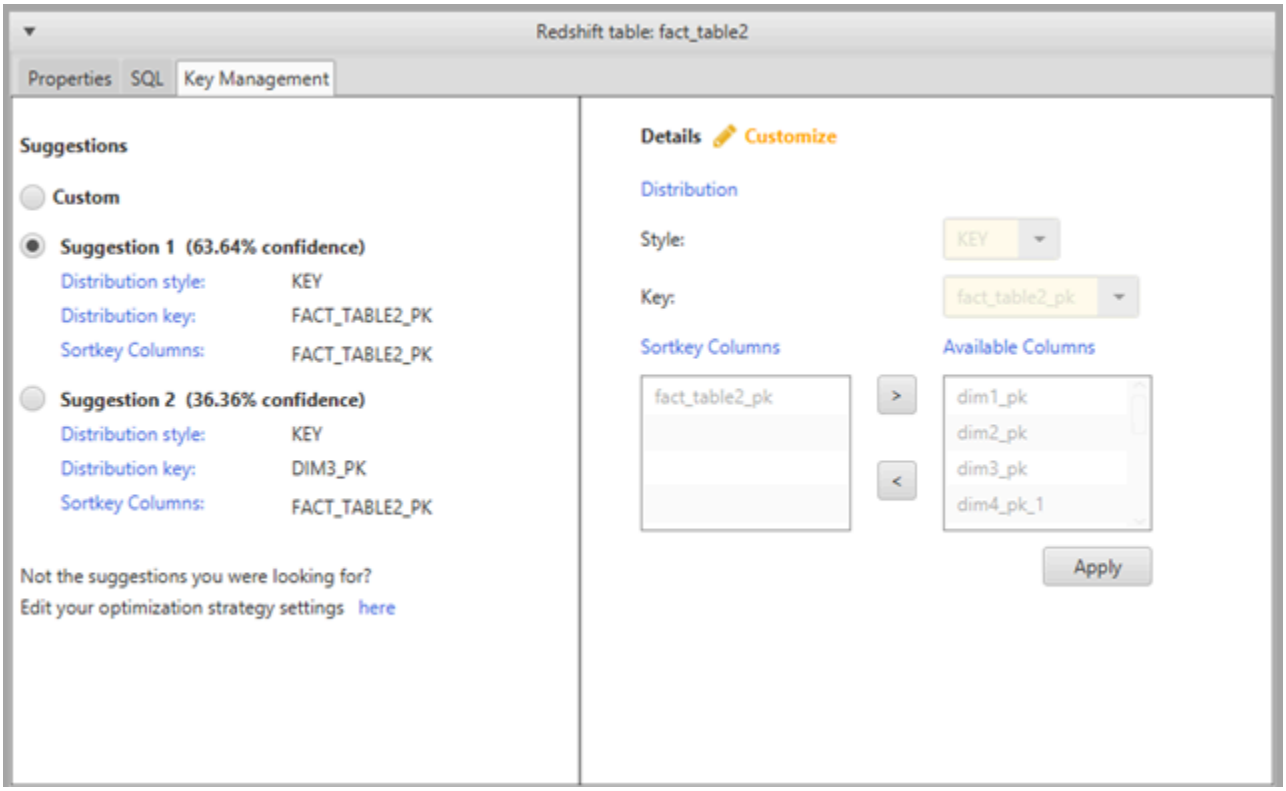
清除转换后的架构

在您将架构应用到目标数据库之前，AWS SCT 仅在本地将转换后的架构存储在项目中。您可以通过选择目标数据库的树状图节点，然后选择 Refresh from Database 来清除您的项目中的计划架构。由于尚未向目标数据库中写入架构，从数据库刷新会删除 AWS SCT 项目中的计划架构元素，以便与目标数据库中的现有元素相匹配。

在 AWS SCT 中管理和自定义键

在使用 AWS Schema Conversion Tool 转换架构后，您可以管理并编辑键。键管理是数据仓库转换的核心。

要管理键，请在目标数据库中选择一个表，然后选择 Key Management 选项卡，如下所示。



左侧窗格包含键建议，以及每个建议的置信评级。您可以选择其中一个建议，也可以通过在右侧窗格中编辑来自定义键。

如果键的选择与预期不相符，您可以编辑优化策略，然后重试转换。有关更多信息，请参阅选择 AWS SCT 所用的优化策略和规则。

相关主题

- 选择最佳的排序键
- 选择最佳的分配方式

在 AWS SCT 中创建和使用评估报告。

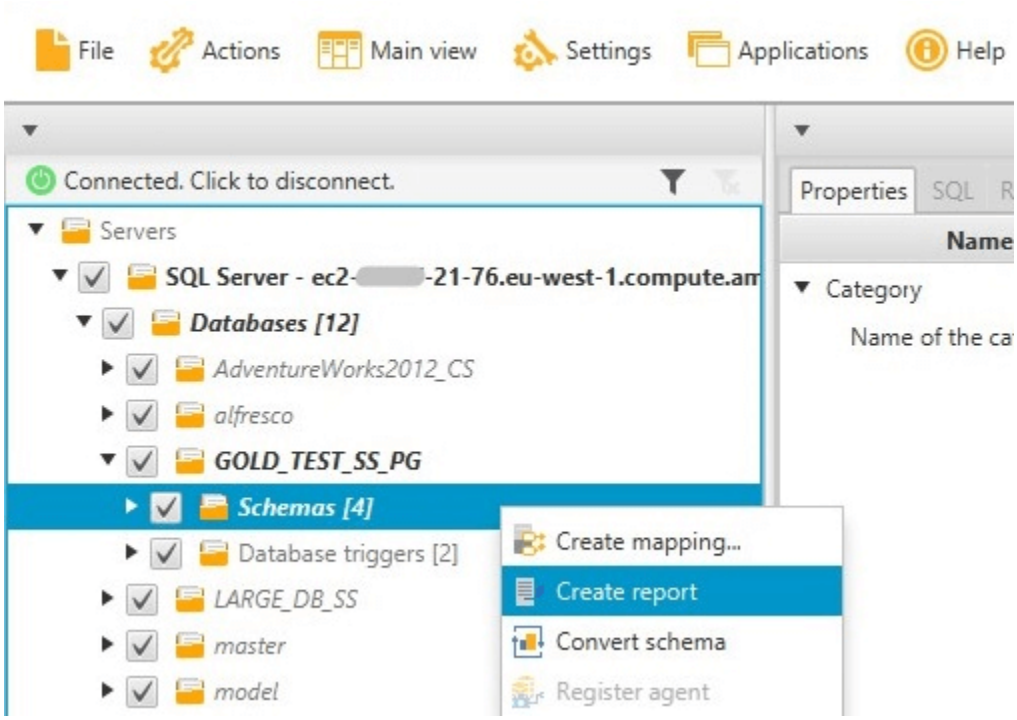
AWS Schema Conversion Tool 会创建数据库迁移评估报告，以帮助您转换架构。数据库迁移评估报告提供了关于将源数据库中的架构转换到目标数据库的重要信息。该报告汇总了所有架构转换任务，针对无法转换到目标数据库的数据库引擎的架构，还详细介绍了其操作项。该报告还包含：对于无法自动转换的部分，在您的目标数据库中编写等效代码所需付出的工作量的估算。

创建数据库迁移评估报告

使用以下过程创建数据库迁移评估报告。

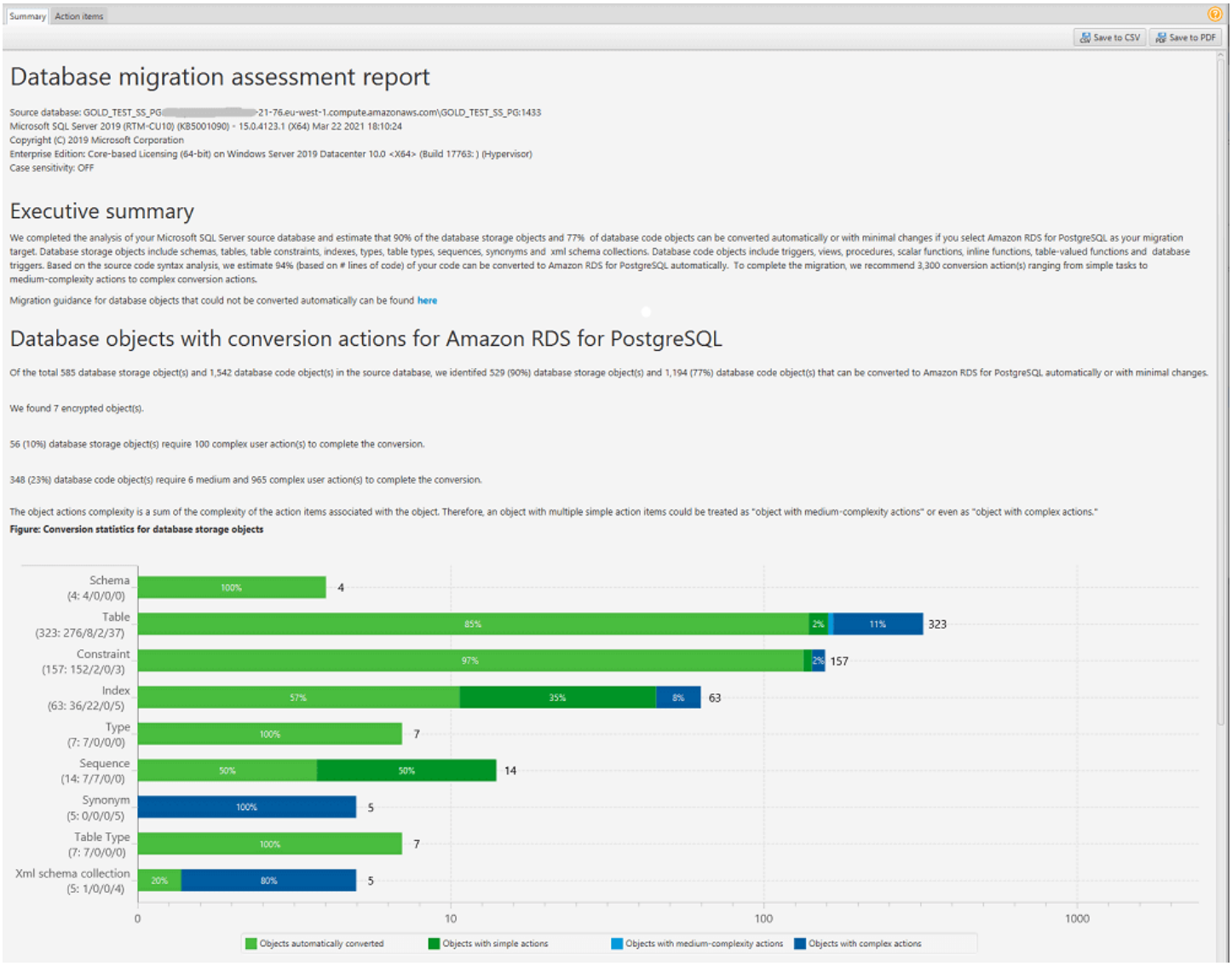
创建数据库迁移评估报告

1. 在显示源数据库架构的左侧面板中，选择要为其创建评估报告的架构对象。
2. 打开该对象的上下文 (右键单击) 菜单，然后选择 Create Report。



评估报告摘要

创建完评估报告后，评估报告视图会打开，其中显示 Summary 选项卡。Summary 选项卡显示了来自数据库迁移评估报告的摘要信息。它显示了已自动转换的项目和未自动转换的项目。



对于无法自动转换为目标数据库引擎的架构项目，摘要包含了在您的目标数据库实例中创建与源数据库中的架构项目等效的架构项目所需的工作量的估算。

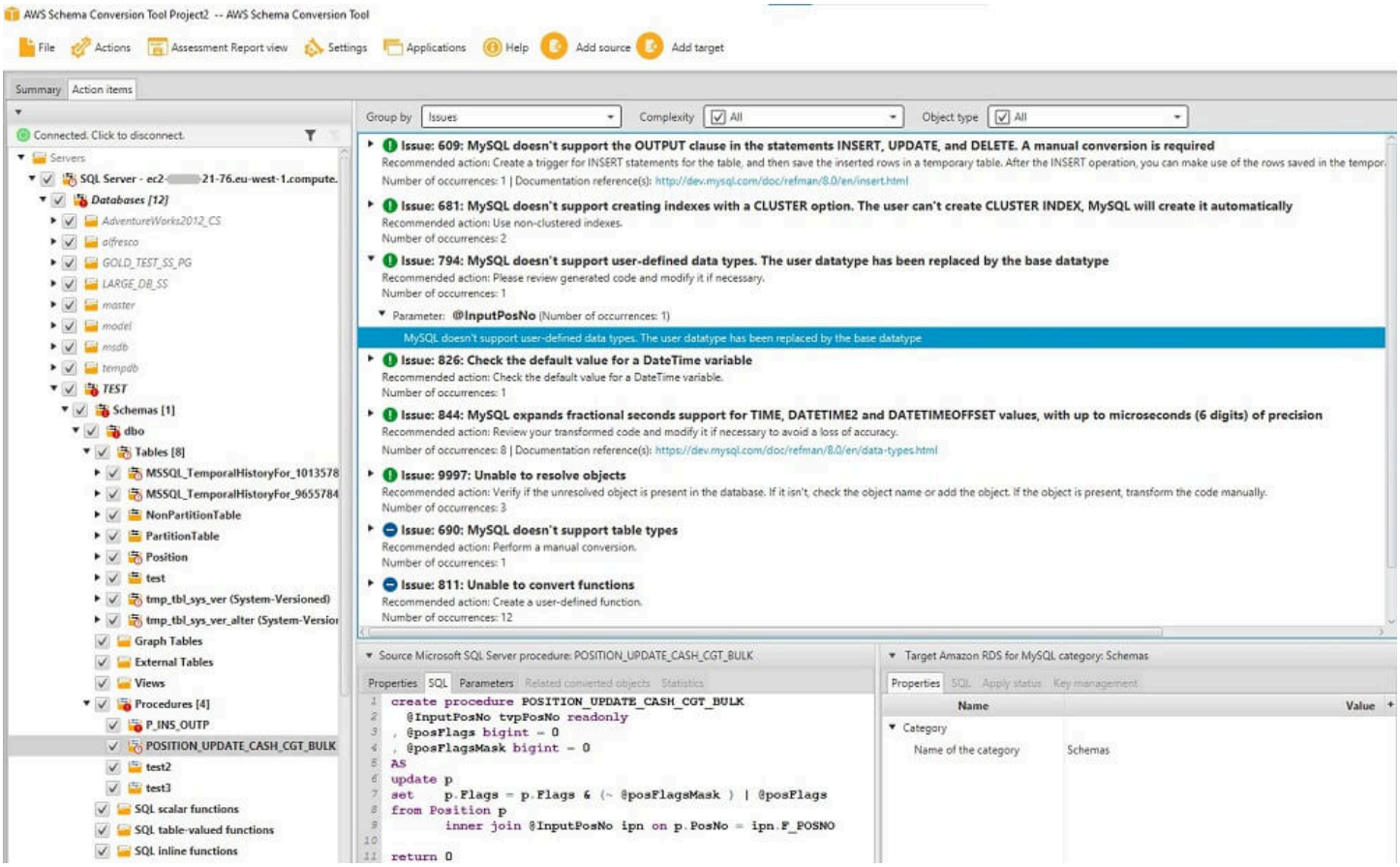
该报告将转换这些架构项目的估算时间划分为以下类别：

- 少量：可在 1 小时之内完成的操作。
- 中度：可在 1 到 4 小时内完成的较为复杂的操作。
- 大量：需要 4 小时以上才能完成的非常复杂的操作。

评估报告操作项

评估报告视图还包含 Action Items 选项卡。此选项卡包含无法自动转换到目标数据库的数据库引擎的项目列表。如果从列表中选择一个操作项，AWS SCT 会突出显示您架构中该操作项适用的项。

该报告还包含有关如何手动转换架构项目的建议。有关如何处理手动转换的更多信息，请参阅在 AWS SCT 中处理手动转换。



保存评估报告

您可以将数据库迁移评估报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。CSV 文件仅包含操作项信息。PDF 文件包含摘要和操作项信息，如以下示例所示。

Database objects with conversion actions for Amazon RDS for PostgreSQL

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

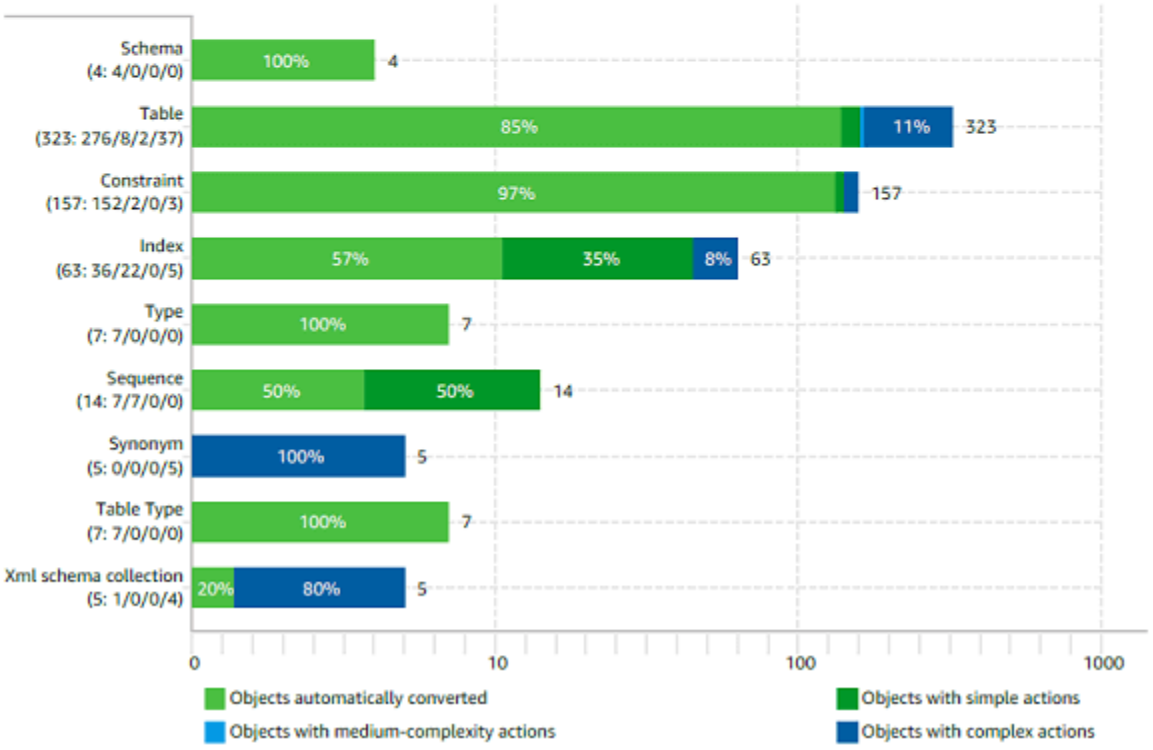
We found 7 encrypted object(s).

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

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

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

Figure: Conversion statistics for database storage objects



在 AWS SCT 中处理手动转换

评估报告包含一系列无法自动转换到目标数据库的数据库引擎的项目。对于无法转换的每一项，Action Items 选项卡上都有一个操作项。

您可以按如下方式应对评估报告中的操作项：

- 修改您的源数据库架构。
- 修改您的目标数据库架构。

修改源架构

对于某些项目，将源数据库中的数据库架构修改为可自动转换的架构可能更容易。首先验证新更改与您的应用程序架构兼容，然后更新源数据库中的架构。最后，用更新的架构信息刷新您的项目。然后，您可以转换更新后的架构，并生成新的数据库迁移评估报告。对于在源架构中更改的项目，不再显示操作项。

此过程的优势是，当您从源数据库刷新时，更新后的架构始终可用。

修改目标架构

对于某些项目，更容易的方法可能是将转换后的架构应用于目标数据库，然后手动将无法自动转换的项目的等效架构项目添加到目标数据库。您可以编写所有可通过应用该架构自动转换到目标数据库的架构。有关更多信息，请参阅保存和应用 AWS SCT 中的转换后的架构。

写入到目标数据库中的架构不包含无法自动转换的项目。在将架构应用到目标数据库后，您就可以在目标数据库中手动创建与源数据库中的架构等效的架构。数据库迁移评估报告中的操作项包含有关如何创建等效架构的建议。

> **Warning**
>
> 如果您在目标数据库中手动创建构架，请保存所做的所有手动操作的副本。如果再次将您的项目的转换后架构应用到目标数据库，它将覆盖您所做的手动操作。

在某些情况下，您无法在目标数据库中创建等效架构。您可能需要重新架构一部分应用程序和数据库，以便将该数据库引擎提供的功能用于您的目标数据库。在其他情况下，您可以简单地忽略无法自动转换的架构。

更新和刷新 AWS SCT 中的转换后的架构

您可以更新 AWS Schema Conversion Tool 项目中的源架构和目标架构。

- 源：如果您更新源数据库的架构，AWS SCT 将用源数据库中的最新架构代替项目中的架构。如果您已对源数据库的架构进行了更改，则可使用此功能来更新项目。
- 目标：如果您更新目标数据库的架构，AWS SCT 将用目标数据库中的最新架构代替您的项目中的架构。如果您尚未将任何架构应用到目标数据库，AWS SCT 将从您的项目中清除转换后的架构。然后，您可以转换源数据库中的架构，以获得干净的目标数据库。

您可以通过选择从数据库刷新更新 AWS SCT 项目中的架构。

保存和应用 AWS SCT 中的转换后的架构

当 AWS Schema Conversion Tool 生成转换后的架构（如 使用 AWS SCT 转换架构 中所示）后，不会立即将转换后的架构应用到目标数据库。而会在本地将转换后的架构存储在项目中，直到您准备好将其应用到目标数据库。使用此功能，您可以使用无法自动转换到目标数据库引擎的架构项目。有关无法自动转换的项目的更多信息，请参阅使用 AWS SCT 创建评估报告。

您可以选择在将架构应用到目标数据库之前，让该工具将转换后的架构作为 SQL 脚本保存到文件中。此外，您还可以让该工具将转换后的架构直接应用到目标数据库。

将转换后的架构保存到文件中

您可以将转换后的架构作为 SQL 脚本保存到一个文本文件中。使用此方法，您可以将 AWS SCT 中生成的 SQL 脚本修改为该工具无法自动转换的地址项。然后，您可以在目标数据库实例上运行更新的脚本，将转换后的架构应用于目标数据库。

将转换后的架构另存为 SQL 脚本

1. 选择架构并打开上下文 (右键单击) 菜单。
2. 选择另存为 SQL。
3. 输入文件名并选择保存。
4. 使用以下选项之一保存转换后的架构：

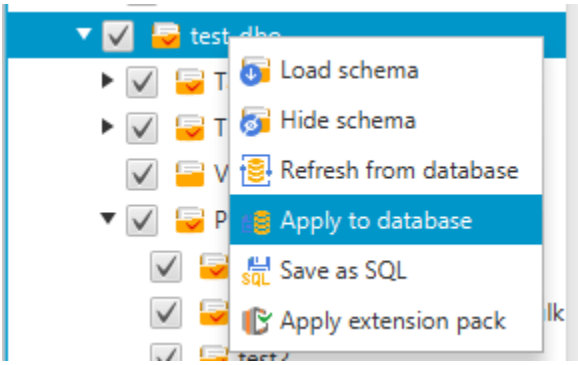
    - 单个文件
    - 每个阶段单个文件
    - 每个语句单个文件

选择 SQL 脚本的格式

1. 在设置菜单上，选择项目设置。
2. 选择保存脚本。
3. 对于供应商，选择数据库平台。
4. 在将 SQL 脚本保存到中，选择数据库架构脚本的保存方式。
5. 选择确定保存设置。

应用转换后的架构

在您准备好将转换后的架构应用到目标数据库后，请从项目的右侧面板中选择该架构元素。打开架构元素的上下文 (右键单击) 菜单，然后选择 Apply to database，如下所示。



扩展包架构

首次将转换后的架构应用于目标数据库实例时，AWS SCT 会向目标数据库实例添加一个额外的架构。该架构用于实现将转换后的架构写入到目标数据库实例时必需的源数据库的系统功能。该架构称为扩展包架构。

不要修改扩展包架构，否则，您可能会在写入到目标数据库实例的转换后架构中遇到意外结果。将架构完全迁移到目标数据库实例后，就不再需要 AWS SCT 了，您可以删除该扩展包架构。

扩展包架构按照您的源数据库命名，如下所示：

- Greenplum：`aws_greenplum_ext`
- Microsoft SQL Server: `aws_sqlserver_ext`
- Netezza：`aws_netezza_ext`
- Oracle：`aws_oracle_ext`
- Snowflake：`aws_snowflake_ext`
- Teradata：`aws_teradata_ext`
- Vertica：`aws_vertica_ext`

有关更多信息，请参阅使用 AWS SCT 扩展包。

Python 库

要在 Amazon Redshift 中创建自定义函数，您可以使用 Python 语言。使用 AWS SCT 扩展包以安装 Amazon Redshift 数据库的 python 库。有关更多信息，请参阅使用 AWS SCT 扩展包。

使用 AWS SCT 优化 Amazon Redshift

您可以使用 AWS Schema Conversion Tool 优化 Amazon Redshift 数据库。使用 Amazon Redshift 数据库作为源，使用测试 Amazon Redshift 数据库作为目标，AWS SCT 建议通过排序键和分配键来优化您的数据库。

优化 Amazon Redshift 数据库

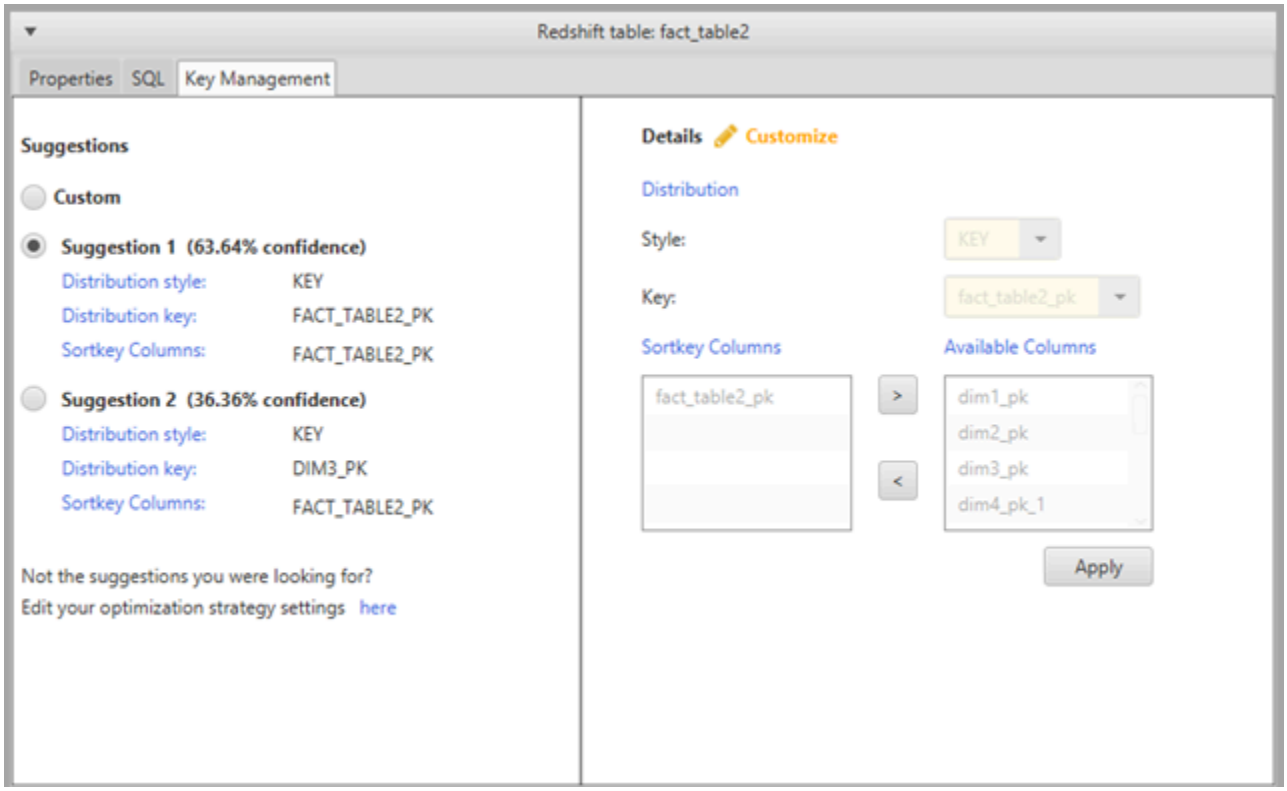
使用以下过程优化您的 Amazon Redshift 数据库。

优化 Amazon Redshift 数据库

1. 为 Amazon Redshift 集群拍摄手动快照以作为备份。优化 Amazon Redshift 集群并对所做的任何更改进行测试后，您可以删除该快照。有关更多信息，请参阅 Amazon Redshift 快照。
2. 从项目左侧面板中选择一个要转换的架构对象。打开该对象的上下文 (右键单击) 菜单，然后选择 Collect Statistics。

   AWS SCT 使用统计数据针对排序键和分配键提出建议。
3. 从项目左侧面板中选择一个要优化的架构对象。打开该对象的上下文 (右键单击) 菜单，然后选择 Run Optimization。

   AWS SCT 针对排序键和分配键提出建议。
4. 要查看建议，请在项目左侧面板的架构下方展开表节点，然后选择一个表。选择 Key Management 选项卡，如下所示。



左侧窗格包含键建议，以及每个建议的置信评级。您可以选择其中一个建议，也可以通过在右侧窗格中编辑来自定义键。

5. 您可以创建一个包含优化建议的报告。要创建该报告，请执行以下操作：

   a. 从项目左侧面板中选择一个已经优化的架构对象。打开该对象的上下文 (右键单击) 菜单，然后选择 Create Report。

      该报告将在主窗口中打开，并显示 Summary 选项卡。报告将显示具有优化建议的对象的数量。

   b. 选择 Action Items 选项卡，以查看报告格式的密钥建议。

   c. 您可以将优化报告的本地副本另存为 PDF 文件或逗号分隔值 (CSV) 文件。CSV 文件仅包含操作项信息。PDF 文件包含摘要和操作项信息。

6. 要将建议的优化应用于数据库，请在项目右侧窗格中选择一个对象。打开对象的上下文 (右键单击) 菜单，然后选择 Apply to database。

使用 AWS Schema Conversion Tool 转换提取、转换和加载（ETL）过程

您可以使用 AWS Schema Conversion Tool（AWS SCT）迁移提取、转换和加载（ETL）过程。这种类型的迁移包括与 ETL 相关的业务逻辑的转换。此逻辑可以位于源数据仓库中，也可以位于您单独运行的外部脚本中。

目前，AWS SCT 支持将 ETL 脚本转换为 AWS Glue 和 Amazon Redshift RSQL 对象，如下表所示。

源	目标
Informatica ETL 脚本	Informatica
Microsoft SQL Server Integration Services (SSIS) ETL 包	AWS Glue 或 AWS Glue Studio
带有来自 Teradata Basic Teradata Query (BTEQ) 的嵌入式命令的 Shell 脚本	Amazon Redshift RSQL
Teradata BTEQ ETL 脚本	AWS Glue 或 Amazon Redshift RSQL
Teradata FastExport 作业脚本	Amazon Redshift RSQL
Teradata FastLoad 作业脚本	Amazon Redshift RSQL
Teradata MultiLoad 作业脚本	Amazon Redshift RSQL

主题

- 使用 AWS SCT 将 Teradata FastExport 作业脚本转换为 Amazon Redshift RSQL
- 使用 AWS SCT 将 Teradata FastLoad 作业脚本转换为 Amazon Redshift RSQL
- 使用 AWS SCT 将 Teradata MultiLoad 作业脚本转换为 Amazon Redshift RSQL

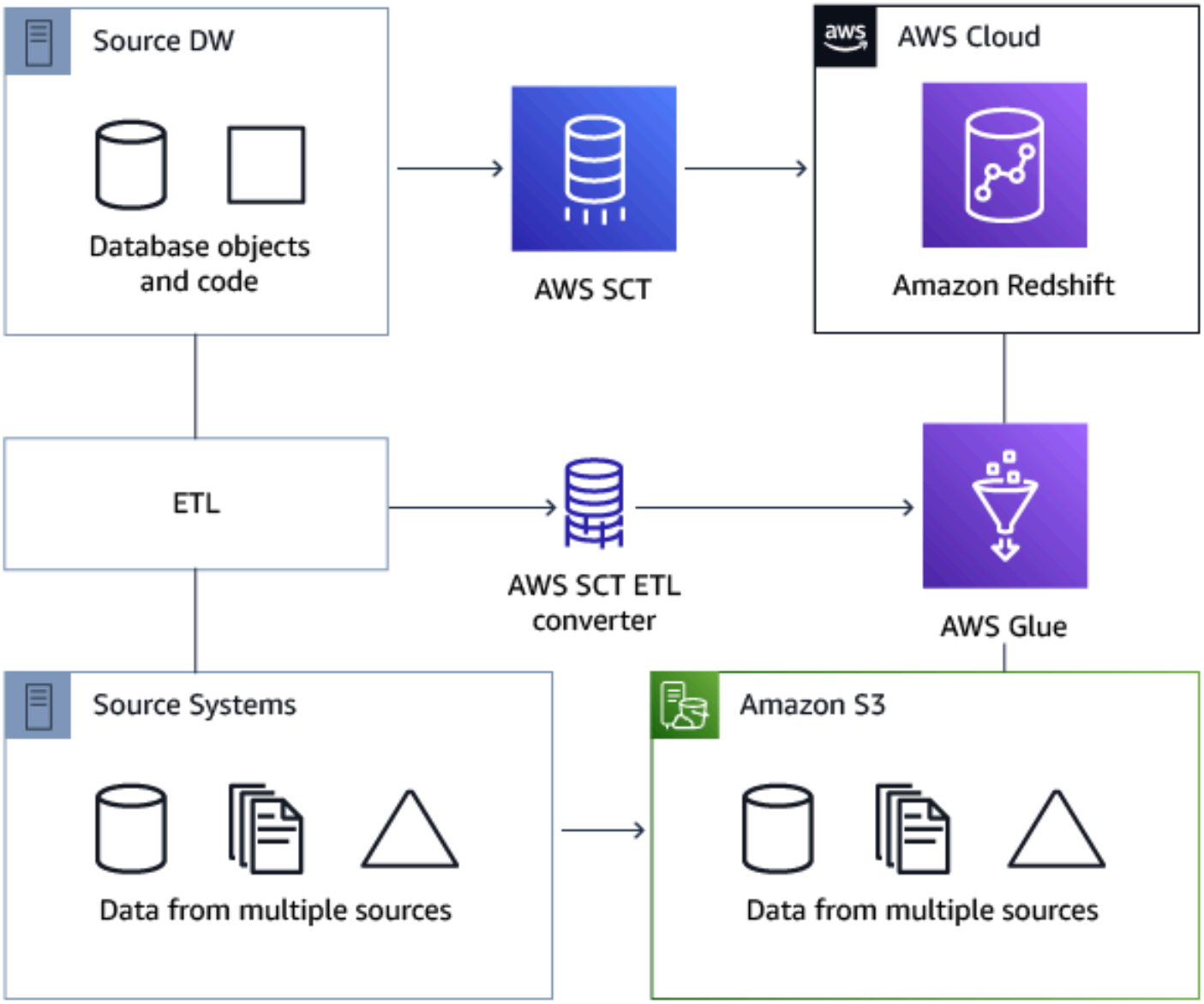
用 AWS SCT 将 ETL 过程转换为 AWS Glue

在以下部分中，您可以找到使用 AWS SCT 将 ETL 脚本转换为 AWS Glue 的过程概述。在本示例中，我们将 Oracle 数据库转换为 Amazon Redshift，并将 ETL 过程与源数据库和数据仓库结合使用。

主题

- 先决条件
- 了解 AWS Glue 数据目录
- 将 AWS SCT 和 AWS Glue 结合使用进行转换的限制
- 步骤 1：创建新项目
- 步骤 2：创建 AWS Glue 作业

以下架构图显示了一个示例数据库迁移项目，其中包括将 ETL 脚本转换为 AWS Glue。



先决条件

开始之前，请执行以下操作：

- 迁移您打算迁移到 AWS 的任何源数据库。
- 将目标数据仓库迁移到 AWS。
- 收集 ETL 过程中涉及的所有代码的列表。
- 收集每个数据库的所有必要连接信息的列表。

此外，AWS Glue 需要权限才能代表您访问其他 AWS 服务。您通过使用 AWS Identity and Access Management（IAM）提供这些权限。请确保您已为 AWS Glue 创建了 IAM 策略。有关更多信息，请参阅《AWS Glue 开发人员指南》中的创建 AWS Glueservice IAM 策略。

了解 AWS Glue 数据目录

作为转换过程的一部分，AWS Glue 加载与源数据库和目标数据库相关的信息。它将这些信息分成不同的类别，并采用称为 树的结构。此结构包括以下内容：

- 连接：连接参数
- 爬网程序：爬网程序的列表，每个架构对应一个爬网程序
- 数据库：容纳表的容器
- 表：表示表中数据的元数据定义
- ETL 作业：执行 ETL 工作的业务逻辑
- 触发器：用于控制 ETL 作业何时在 AWS Glue 中运行（是否按需、按计划或由作业事件触发）的逻辑

AWS Glue 数据目录 是数据的位置、架构和运行时指标的索引。当您结合使用 AWS Glue 与 AWS SCT 时，AWS Glue 数据目录包含对在 AWS Glue 中用作 ETL 作业的源和目标的数据的引用。要创建数据仓库，请编录该数据。

您可以使用数据目录中的信息创建和监控您的 ETL 作业。通常，您运行爬网程序来清点数据存储中的数据，但还有其他方法可以将元数据表添加到数据目录中。

当您在数据目录中定义表时，您将其添加到数据库中。数据库用于组织 AWS Glue 中的表。

将 AWS SCT 和 AWS Glue 结合使用进行转换的限制

结合使用 AWS SCT 与 AWS Glue 进行转换时存在以下限制。

资源	默认限制
每个账户的数据库数量	10000
每个数据库的表数量	100000
每个表的分区数量	1000000

每个表的表版本数量	100000
每个账户的表数量	1000000
每个账户的分区数量	10,000,000
每个账户的表版本数量	1000000
每个账户的连接数量	1000
每个账户的爬网程序数量	25
每个账户的作业数量	25
每个账户的触发器数量	25
每个账户的并发作业运行数量	30
每个作业的并发作业运行数量	3
每个触发器的作业数量	10
每个账户的开发端点数量	5
开发端点一次使用的最大数据处理单元（DPU）数	5
角色一次使用的最大 DPU 数	100
数据库名称长度	无限 为了与其他元数据存储（如 Apache Hive）兼容，名称会更改为使用小写字符。 如果您计划从 Amazon Athena 访问数据库，请提供只包含字母数字和下划线字符的名称。
连接名称长度	无限
爬网程序名称长度	无限

步骤 1：创建新项目

要创建新项目，请执行以下简要步骤：

1. 在 AWS SCT 中创建一个新项目。有关更多信息，请参阅创建 AWS SCT 项目。
2. 将源数据库和目标数据库添加到项目中。有关更多信息，请参阅向 AWS SCT 项目添加数据库服务器。

   确保已在目标数据库连接设置中选择了使用 AWS Glue。为此，请选择 AWS Glue 选项卡。对于从 AWS 配置文件复制，选择要使用的配置文件。配置文件应自动填写 AWS 访问密钥、私有密钥和 Amazon S3 存储桶文件夹。如果不是这样，请自行输入这些信息。选择 OK (确定) 后，AWS Glue 分析对象并将元数据加载到 AWS Glue 数据目录中。

   根据您的安全设置，您可能会收到一条警告消息，指出您的账户对服务器上的某些架构没有足够的权限。如果您有权访问您使用的架构，您可以安全地忽略此消息。
3. 要完成准备导入您的 ETL，请连接到源数据库和目标数据库。为此，请在源元数据树或目标元数据树中选择您的数据库，然后选择连接到服务器。

AWS Glue 在源数据库服务器上创建一个数据库，也在目标数据库服务器创建一个数据库，以帮助进行 ETL 转换。目标服务器上的数据库包含 AWS Glue 数据目录。要查找特定对象，请在源面板或目标面板上使用搜索。

要查看特定对象如何转换，请找到要转换的项，然后从其上下文（右键单击）菜单中选择转换架构。AWS SCT 将此选定的对象转换为脚本。

您可以在右侧面板的脚本文件夹中查看转换后的脚本。当前，脚本是一个虚拟对象，只能作为 AWS SCT 项目的一部分使用。

要使用转换后的脚本创建 AWS Glue 作业，请将脚本上传到 Amazon S3。要将脚本上传到 Amazon S3，请选择脚本，然后从其上下文（右键单击）菜单中选择 保存至 S3。

步骤 2：创建 AWS Glue 作业

将脚本保存到 Amazon S3 后，您可以选择它，然后选择配置 AWS Glue 作业以打开向导配置 AWS Glue 作业。此向导可让您更轻松地对其进行设置：

1. 在向导的第一个选项卡设计数据流上，您可以选择执行策略以及要纳入这一个作业的脚本列表。您可以为每个脚本选择参数。也可以重新排列脚本，以便它们以正确的顺序运行。

2. 在第二个选项卡上，您可以为作业命名，并直接配置 AWS Glue 的设置。在此屏幕上，您可以配置以下设置：
   - AWS Identity and Access Management（IAM）角色
   - 脚本文件名和文件路径
   - 使用具有 Amazon S3 托管式密钥的服务器端加密（SSE-S3）加密脚本
   - 临时目录
   - 生成的 Python 库路径
   - 用户 Python 库路径
   - 从属 .jar 文件的路径
   - 引用的文件路径
   - 每个作业运行的并发 DPU 数量
   - 最大并发数量
   - 作业超时（分钟）
   - 延迟通知阈值（分钟）
   - 重试次数
   - 安全配置
   - 服务器端加密
3. 在第三个步骤或选项卡中，您可以选择已配置的到目标端点的连接。

配置完作业后，作业会显示在 AWS Glue 数据目录中的 ETL 作业下。如果您选择该作业，设置将显示，因此您可以查看或编辑它们。要在 AWS Glue 中创建新的作业，请从作业的上下文（右键单击）菜单中选择创建 AWS Glue 作业。执行此操作将应用架构定义。要刷新显示内容，请从上下文（右键单击）菜单中选择 Refresh from database (从数据库刷新)。

此时，您可以在 AWS Glue 控制台中查看您的作业。为此，请登录 AWS Management Console 并通过以下网址打开 AWS Glue 控制台：https://console.aws.amazon.com/glue/。

您可以测试新作业以确保它正确工作。为此，首先检查源表中的数据，然后验证目标表是否为空。运行作业，然后再次检查。可以从 AWS Glue 控制台中查看错误日志。

将适用于 AWS Glue 的 Python API 与 AWS SCT 配合使用转换 ETL 过程

在以下部分中，您可以找到在 Python 中调用 AWS Glue API 操作的转换的描述。有关更多信息，请参阅《AWS Glue 开发人员指南》中的使用 Python 编程 AWS Glue ETL 脚本。

主题

步骤 1：创建数据库

第一步是使用 AWS SDK API 在 AWS Glue 数据目录中创建一个新的数据库。当您在数据目录中定义表时，您将其添加到数据库。数据库用于组织 AWS Glue 中的表。

以下示例演示了适用于 AWS Glue 的 Python API 的 `create_database` 方法。

```
response = client.create_database(
    DatabaseInput={
        'Name': 'database_name',
        'Description': 'description',
        'LocationUri': 'string',
        'Parameters': {
            'parameter-name': 'parameter value'
        }
    }
)
```

如果您使用的是 Amazon Redshift，数据库名称的构成方式如下。

```
{redshift_cluster_name}_{redshift_database_name}_{redshift_schema_name}
```

此示例中的 Amazon Redshift 集群的完整名称如下所示。

```
rsdbb03.apq1mpqso.us-west-2.redshift.amazonaws.com
```

下面显示了格式正确的数据库名称的示例。在此例中，rsdbb03 为名称，这是集群端点的完整名称的第一个部分。数据库名为 dev，架构为 ora_glue。

```
rsdbb03_dev_ora_glue
```

步骤 2：创建连接

使用 AWS SDK API 在数据目录中创建新连接。

以下示例演示了适用于 AWS Glue 的 Python API 的 create_connection 方法。

```
response = client.create_connection(
    ConnectionInput={
        'Name': 'Redshift_abcde03.aabbcc112233.us-west-2.redshift.amazonaws.com_dev',
        'Description': 'Created from SCT',
        'ConnectionType': 'JDBC',
        'ConnectionProperties': {
            'JDBC_CONNECTION_URL': 'jdbc:redshift://aabbcc03.aabbcc112233.us-
west-2.redshift.amazonaws.com:5439/dev',
            'USERNAME': 'user_name',
            'PASSWORD': 'password'
        },
        'PhysicalConnectionRequirements': {
            'AvailabilityZone': 'us-west-2c',
            'SubnetId': 'subnet-a1b23c45',
            'SecurityGroupIdList': [
                'sg-000a2b3c', 'sg-1a230b4c', 'sg-aba12c3d', 'sg-1abb2345'
            ]
        }
    }
)
```

create_connection 中所用的参数如下所示：

- Name（UTF-8 字符串）：必需。对于 Amazon Redshift，连接名称的构成方式如下所示：Redshift_*<Endpoint-name>*_*<redshift-database-name>*，例如：Redshift_abcde03_dev
- Description（UTF-8 字符串）：连接的描述。
- ConnectionType（UTF-8 字符串）：必需。连接的类型。当前，仅支持 JDBC；SFTP 不受支持。

- ConnectionProperties（dict）：必需。用作此连接的参数的键值对列表，包括 JDBC 连接 URL、用户名和密码。
- PhysicalConnectionRequirements（dict）：物理连接要求，其中包括以下内容：
  - SubnetId（UTF-8 字符串）：连接使用的子网的 ID。
  - SecurityGroupIdList（列表）：连接使用的安全组 ID 列表。
  - AvailabilityZone（UTF-8 字符串）：必需。包含该端点的可用区。此参数已被弃用。

步骤 3：创建 AWS Glue 爬网程序

接下来，您将创建一个 AWS Glue 爬网程序来填充 AWS Glue 目录。有关更多信息，请参阅《AWS Glue 开发人员指南》中的使用爬网程序编录表。

添加爬网程序的第一步是使用 AWS SDK API 在数据目录中创建新数据库。在开始之前，务必先使用 delete_crawler 操作删除其先前的任何版本。

创建爬网程序时，请注意以下几点：

- 对于爬网程序名称，使用格式 *<redshift_node_name>*_*<redshift_database_name>*_*<redshift_shema_name>*，例如：abcde03_dev_ora_glue
- 使用已存在的 IAM 角色。有关创建 IAM 角色的更多信息，请参阅《IAM 用户指南》中的创建 IAM 角色。
- 请使用您在先前步骤中创建的数据库的名称。
- 使用 ConnectionName 参数，这是必需的。
- 对于 path 参数，使用 JDBC 目标的路径，例如：dev/ora_glue/%

以下示例将删除现有爬网程序，然后使用适用于 AWS Glue 的 Python API 创建新的爬网程序。

```
response = client.delete_crawler(
    Name='crawler_name'
)

response = client.create_crawler(
    Name='crawler_name',
    Role= 'IAM_role',
    DatabaseName='database_name',
    Description='string',
```

```
    Targets={
        'S3Targets': [
            {
                'Path': 'string',
                'Exclusions': [
                    'string',
                ]
            },
        ],
        'JdbcTargets': [
            {
                'ConnectionName': 'ConnectionName',
                'Path': 'Include_path',
                'Exclusions': [
                    'string',
                ]
            },
        ]
    },
    Schedule='string',
    Classifiers=[
        'string',
    ],
    TablePrefix='string',
    SchemaChangePolicy={
        'UpdateBehavior': 'LOG'|'UPDATE_IN_DATABASE',
        'DeleteBehavior': 'LOG'|'DELETE_FROM_DATABASE'|'DEPRECATE_IN_DATABASE'
    },
    Configuration='string'
)
```

创建并运行一个爬网程序，以便连接到一个或多个数据存储，确定数据结构，并将表写入到数据目录中。您可以按计划运行您的爬网程序，如下所示。

```
response = client.start_crawler(
    Name='string'
)
```

此示例使用 Amazon Redshift 作为目标。在爬网程序执行后，Amazon Redshift 数据类型按以下方式映射到 AWS Glue 数据类型：

Amazon Redshift 数据类型	AWS Glue 数据类型

smallint	smallint
integer	int
bigint	bigint
decimal	decimal(18,0)
decimal(p,s)	decimal(p,s)
real	double
double precision	double
布尔值	布尔值
char	字符串
varchar	字符串
varchar(n)	字符串
date	date
timestamp	timestamp
timestamptz	timestamp

使用 AWS SCT 转换 Informatica ETL 脚本

您可以使用 AWS SCT 命令行界面（CLI）转换 Informatica ETL 脚本，以便可以在新的目标数据库中使用这些脚本。此转换包括三个关键步骤。首先，AWS SCT 转换嵌入在 Informatica 对象中的 SQL 代码。接下来，AWS SCT 会根据您在项目中指定的迁移规则更改数据库对象的名称。最后，AWS SCT 会将 Informatica ETL 脚本的连接重定向到新的目标数据库。

您可以将 Informatica ETL 脚本作为 AWS SCT 数据库转换项目的一部分进行转换。在转换 Informatica ETL 脚本时，请务必将源数据库和目标数据库添加到项目中。

要转换 Informatica ETL 脚本，请确保使用 AWS SCT 版本 1.0.667 或更高版本。另外，请熟悉 AWS SCT 的命令行界面。有关更多信息，请参阅AWS SCT CLI 参考。

使用 AWS SCT 转换 Informatica ETL 脚本

1. 创建新的 AWS SCT CLI 脚本或编辑现有场景模板。例如，您可以下载和编辑 `InformaticConversionTemplate.scts` 模板。有关更多信息，请参阅获取 CLI 场景。
2. 下载源数据库和目标数据库所需的 JDBC 驱动程序。使用 `SetGlobalSettings` 命令指定这些驱动程序的位置。此外，还要指定 AWS SCT 可以保存日志文件的文件夹。

   以下代码示例向您展示了如何将 Oracle 和 PostgreSQL 驱动程序的路径添加到 AWS SCT 设置中。运行此代码示例后，AWS SCT 将日志文件存储在 `C:\sct_log` 文件夹中。此外，AWS SCT 还会将控制台日志文件存储在 `C:\Temp\oracle_postgresql` 文件夹中。

   ```
   SetGlobalSettings
    -save: 'true'
    -settings: '{"oracle_driver_file": "C:\\drivers\\ojdbc8.jar",
    "postgresql_driver_file": "C:\\drivers\\postgresql-42.2.19.jar" }'
   /

   SetGlobalSettings
    -save: 'false'
    -settings: '{
   "log_folder": "C:\\sct_log",
   "console_log_folder": "C:\\Temp\\oracle_postgresql"}'
   /
   ```
3. 创建新 AWS SCT 项目。输入项目的名称和位置。

   以下代码示例在 `C:\Temp` 文件夹中创建 `oracle_postgresql` 项目。

   ```
   CreateProject
    -name: 'oracle_postgresql'
    -directory: 'C:\Temp'
   /
   ```
4. 添加有关源数据库和目标数据库的连接信息。

   以下代码示例将 Oracle 和 PostgreSQL 数据库添加为 AWS SCT 项目的源和目标。

   ```
   AddSource
    -password: 'source_password'
    -port: '1521'
    -vendor: 'ORACLE'
    -name: 'ORACLE'
   ```

```
 -host: 'source_address'
 -database: 'ORCL'
 -user: 'source_user'
/
AddTarget
 -database: 'postgresql'
 -password: 'target_password'
 -port: '5432'
 -vendor: 'POSTGRESQL'
 -name: 'POSTGRESQL'
 -host: 'target_address'
 -user: 'target_user'
/
```

在前面的示例中，将 *source_user* 和 *target_user* 替换为数据库用户的名称。接下来，用您的密码替换 *source_password* 和 *target_password*。在 *source_address* 和 *target_address* 中，输入源数据库服务器和目标数据库服务器的 IP 地址。

要连接到版本 19 及更高版本的 Oracle 数据库，请在 AddSource 命令中使用 Oracle 服务名称。为此，请添加 -connectionType 参数并将其值设置为 'basic_service_name'。然后，添加 -servicename 参数并将其值设置为 Oracle 服务名称。有关 AddSource 命令的更多信息，请参阅《AWS Schema Conversion Tool CLI 参考》https://s3.amazonaws.com/publicsctdownload/AWS+SCT+CLI+Reference.pdf。

5. 创建新的 AWS SCT 映射规则，该规则定义每个源数据库架构的目标数据库引擎。有关更多信息，请参阅在 AWS SCT 中创建映射规则。

   以下代码示例创建了一个包含所有源 Oracle 数据库架构的映射规则，并将 PostgreSQL 定义为迁移目标。

   ```
   AddServerMapping
    -sourceTreePath: 'Servers.ORACLE'
    -targetTreePath: 'Servers.POSTGRESQL'
   /
   ```

6. 添加有关 Informatica 源文件和目标 XML 文件的连接信息。

   以下代码示例添加了 C:\Informatica_source 和 C:\Informatica_target 文件夹中的 Informatica XML 文件。

   ```
   AddSource
    -name: 'INFA_SOURCE'
   ```

```
 -vendor: 'INFORMATICA'
 -mappingsFolder: 'C:\Informatica_source'
/
AddTarget
 -name: 'INFA_TARGET'
 -vendor: 'INFORMATICA'
 -mappingsFolder: 'C:\Informatica_target'
/
```

7. 创建另一条映射规则，为源 Informatica XML 文件定义目标 Informatica XML 文件。

   以下代码示例创建了一个映射规则，该规则包括前面示例中使用的源 Informatica XML 文件和目标 Informatica XML 文件。

```
AddServerMapping
-sourceTreePath: 'ETL.INFA_SOURCE'
-targetTreePath: 'ETL.INFA_TARGET'
/
```

8. 指定与 Informatica 连接名称引用对应的数据库服务器连接。

   以下代码示例配置了将 Informatica ETL 脚本从源数据库重定向到新目标数据库的过程。此示例还配置了连接变量。

```
ConfigureInformaticaConnectionsRedirect
 -treePath: 'ETL.INFA_SOURCE.Files'
 -connections: '{
 "ConnectionNames": [
 {
  "name": "Oracle_src",
  "newName": "postgres",
  "treePath": "Servers.ORACLE"
 }
 ]
 "ConnectionVariables": [
 {
         "name": "$Source",
         "treePath": "Servers.ORACLE"
    }
    ]
 }'
/
```

9. 转换源数据库架构和 Informatica ETL 脚本。

   以下代码示例转换您的所有源 Oracle 数据库架构和 Informatica XML 文件。

```
Convert
 -treePath: 'Servers.ORACLE.Schemas.%'
/
Convert
 -treePath: 'ETL.INFA_SOURCE.Files'
/
```

10. （可选）保存转换项目和评估报告。该报告包括转换操作项以及有关如何解决每个问题的建议。

    以下代码示例保存您的项目，并将评估报告的副本作为 PDF 文件保存到 C:\Temp 文件夹中。

```
SaveProject
/
SaveReportPDF
 -treePath: 'ETL.INFA_SOURCE.Files'
 -file:'C:\Temp\Informatica.pdf'
/
```

11. 保存转换后的 Informatica XML 文件。

    以下代码示例将转换后的 XML 文件保存在 C:\Temp 文件夹中。您在上一步中使用 AddTarget 命令指定了此文件夹。

```
SaveTargetInformaticaXML
-treePath: 'ETL.INFA_TARGET.Files'
/
```

12. 将您的脚本保存为 .scts 文件，然后使用 AWS SCT CLI 中的 RunSCTBatch 命令运行它。有关更多信息，请参阅AWS SCT CLI 脚本模式。

    以下示例运行 C:\Temp 文件夹中的 Informatica.scts 脚本。你可以在 Windows 中使用这个示例。

```
RunSCTBatch.cmd --pathtoscts "C:\Temp\Informatica.scts"
```

    如果编辑源 Informatica ETL 脚本，请再次运行 AWS SCT CLI 脚本。

用 AWS SCT 将 SSIS 转换为 AWS Glue

接下来，你可以找到如何使用 AWS SCT 将 Microsoft SQL Server Integration Services (SSIS) 包转换为 AWS Glue。

要将 Microsoft SSIS 包转换为 AWS Glue，请确保使用 AWS SCT 版本 1.0.642 或更高版本。您还需要有一个 SSIS 项目，其中包含 ETL 包 — 本地文件夹中的 `.dtsx`、`.conmgr` 和 `.params` 文件。

您不需要安装 SSIS 服务器。通过本地 SSIS 文件进行转换过程。

使用 AWS SCT 将 SSIS 包转换为 AWS Glue

1. 在 AWS SCT 中创建新项目，或打开现有项目。有关更多信息，请参阅the section called “创建项目”。
2. 在菜单上选择添加源，将新的源 SSIS 包添加到您的项目中。
3. 选择 SQL Server Integration Services 并完成以下操作：

   - 连接名称：输入连接的名称。AWS SCT 在元数据树中显示此名称。
   - SSIS 包文件夹：选择包含包的 SSIS 项目文件夹的路径。

   AWS SCT 从本地文件夹读取项目文件（扩展名 `.dtsx`、`.conmgr` 或 `.params` 的文件）并对其进行解析。然后，它会将它们组织成 AWS SCT 类别树。
4. 从菜单中选择添加目标以添加新的目标平台，从而转换源 SSIS 包。
5. 选择 AWS Glue 并完成以下操作：

   - 连接名称：输入连接的名称。AWS SCT 在元数据树中显示此名称。
   - 从 AWS 配置文件复制：选择要使用的配置文件。
   - AWS 访问密钥：输入您的 AWS 访问密钥。
   - AWS 私有密钥：输入您的 AWS 私有密钥。
   - 区域：从列表中选择要使用的 AWS 区域。
   - Amazon S3 存储桶文件夹：输入您计划使用的 Amazon S3 存储桶的文件夹路径。

   你可以使用虚拟 AWS Glue 目标。在这种情况下，不需要指定连接凭证。有关更多信息，请参阅the section called “虚拟目标”。
6. 创建新的映射规则，其中包含源 SSIS 包和 AWS Glue 目标。有关更多信息，请参阅the section called “新规则”。

7. 在视图菜单上，选择主视图。
8. 在 SSIS 树视图中，打开连接管理器的上下文（右键单击）菜单，然后选择配置连接。
9. 配置项目连接管理器。

   要为 SSIS 连接管理器配置连接映射，请为相应的 SSIS 连接管理器指定 AWS Glue 连接。确保您已创建 AWS Glue 连接。

   a. 在连接下，选择项目连接。
   b. 对于 Glue 目录连接，请选择相应的 AWS Glue 连接。
10. 配置包连接管理器：

    a. 在连接下，选择包。
    b. 对于 Glue 目录连接，请选择相应的 AWS Glue 连接。
    c. 对包可用的所有连接重复这些操作。
11. 选择 Apply（应用）。
12. 转换包。在源树视图中，找到包。打开包的上下文（右键单击）菜单，然后选择转换包。
13. 将转换后的脚本保存到 Amazon S3。在目标树视图中，找到包脚本。打开转换后的脚本的上下文（右键单击）菜单，然后选择保存到 S3。
14. 配置 AWS Glue 作业。在目标树视图中，找到包脚本。打开转换后的脚本的上下文（右键单击）菜单，然后选择配置 AWS Glue 作业。
15. 完成三个配置部分。

    a. 完成设计数据流部分：

       - 执行策略：选择作业将如何运行 ETL 脚本。选择 SEQUENTIAL，按向导中指定的顺序运行脚本。选择 PARALLEL 并行运行脚本，而不考虑向导中指定的顺序。
       - 脚本：选择转换后的脚本的名称。
       - 选择 Next（下一步）。

    b. 完成作业属性部分：

       - 名称：输入 AWS Glue 作业的名称。
       - IAM 角色：选择用于运行作业和访问数据存储的资源进行授权的 IAM 角色。
       - 脚本文件名：输入转换后的脚本的名称。
       - 脚本文件 S3 路径：输入转换后的脚本的 Amazon S3 路径。

- 使用 SSE-S3 加密脚本：选择此选项以使用采用 Amazon S3 托管式加密密钥的服务器端加密（SSE-S3）保护数据。
- 临时目录：输入临时目录的 Amazon S3 路径以获得中间结果。AWS Glue 和 AWS Glue 内置转换则使用此目录读取或写入 Amazon Redshift。
- AWS SCT 会自动生成 Python 库的路径。你可以在生成的 python 库路径中查看此路径。您不能编辑这个自动生成的路径。要使用其他 Python 库，请在用户 python 库路径中输入路径。
- 用户 python 库路径：输入其他用户 Python 库的路径。使用逗号分隔 Amazon S3 路径。
- 从属 JAR 路径：输入从属 JAR 文件的路径。使用逗号分隔 Amazon S3 路径。
- 引用文件路径：输入脚本所需的其他文件（例如配置文件）的路径。使用逗号分隔 Amazon S3 路径。
- 最大容量：输入此作业运行时可分配的 AWS Glue 数据处理单元 (DPU) 的最大数量。可以输入 2 到 100 的任何整数。默认值为 2。
- 最大并发：输入此作业允许的并发运行的最大数量。默认值是 1。达到此阈值时，AWS Glue 将返回一个错误。
- 作业超时（分钟）：输入 ETL 作业的超时值，以防作业失控。批处理作业的默认值为 2880 分钟（48 小时）。当作业执行时间超过此限制时，作业运行状态更改为 TIMEOUT。
- 延迟通知阈值（分钟）：输入 AWS SCT 发送延迟通知之前的阈值（以分钟为单位）。
- 重试次数：输入在失败时 AWS Glue 自动重新启动作业的次数（0–10）。达到超时限制的作业不会重新启动。默认值为 0。
- 选择 Next（下一步）。

c. 配置所需的连接：

i. 从所有连接中，选择所需的 AWS Glue 连接并将其添加到选定连接列表中。

ii. 选择 Finish (结束)。

16. 创建已配置的 AWS Glue 作业。在目标树视图中，查找并展开 ETL 作业。打开您配置的 ETL 作业的上下文（右键单击）菜单，然后选择创建 AWS Glue 作业。
17. 运行 AWS Glue 作业：

a. 打开 AWS Glue 控制台，地址：https://console.aws.amazon.com/glue/。

b. 在导航窗格中，选择作业。

c. 选择添加作业，然后选择要运行的作业。

d. 在操作选项卡上，选择运行作业。

AWS SCT 可以转换为 AWS Glue 的 SSIS 组件

您可以使用 AWS SCT 转换数据流和控制流组件以及容器、参数和变量。

支持的数据流组件包括：

- ADO NET 目的地
- ADO NET 源
- 聚合
- 缓存转换
- 字符映射转换
- 条件拆分转换
- 复制列转换
- 数据转换的转换
- 派生列转换
- Excel 目的地
- Excel 源
- 导出列转换
- 平面文件目的地
- 平面文件源
- 模糊查找转换
- 导入列转换
- 查找转换
- 合并联接转换
- 合并转换
- 组播转换
- ODBC 目的地
- ODBC 源
- OLE 数据库命令转换
- OLE 数据库目标

- OLE DB 源
- 百分比抽样转换
- 转置转换
- 原始文件目的地
- 原始文件源
- RecordSet 目的地
- 行数转换
- 行采样转换
- 排序转换
- SQL Server 目的地
- Union All 转换
- 反转置转换
- XML 源

支持的控制流组件包括：

- 批量插入任务
- 执行包任务
- 执行 SQL 任务
- 执行 T-SQL 语句任务
- 表达式任务
- 文件系统任务
- 通知操作员任务
- 发送邮件任务

支持的 SSIS 容器包括：

- For 循环容器
- Foreach 循环容器
- 顺序容器

用 AWS SCT 将 SSIS 转换为 AWS Glue Studio

你可以使用 AWS SCT 将 Microsoft SQL Server Integration Services (SSIS) 包转换为 AWS Glue Studio。

SSIS 包涵盖运行特定提取、转换和加载 (ETL) 任务所需的必要组件，例如连接管理器、任务、控制流、数据流、参数、事件处理程序和变量。AWS SCT 将 SSIS 包转换为与 AWS Glue Studio 兼容的格式。将源数据库迁移到 AWS Cloud 之后，您可以运行这些转换后的 AWS Glue Studio 作业执行 ETL 任务。

要将 Microsoft SSIS 包转换为 AWS Glue Studio，请确保使用 AWS SCT 版本 1.0.661 或更高版本。

主题

先决条件

在本节中，了解将 SSIS 包转换为 AWS Glue的先决条件任务。这些任务包括在账户中创建所需的 AWS 资源。

您可使用 AWS Identity and Access Management（IAM）定义访问 AWS Glue Studio 使用的资源所需的策略和角色。有关更多信息，请参阅AWS Glue Studio 用户 IAM 权限。

AWS SCT 将源脚本转换为 AWS Glue Studio 后，将转换后的脚本上传到 Amazon S3 存储桶。请务必创建此 Amazon S3 存储桶，并在 AWS 服务配置文件设置中选择它。有关创建 Amazon S3 存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的创建第一个 S3 存储桶。

要确保 AWS Glue Studio 可以连接到数据存储，请创建一个自定义连接器和一个连接。此外，还要将数据确保使用值替换库凭证存储在 AWS Secrets Manager 中。

创建自定义连接器

1. 下载数据存储的 JDBC 驱动程序。有关 AWS SCT 使用的 JDBC 驱动程序的更多信息，请参阅 下载所需的数据库驱动程序。
2. 将驱动程序文件上传到 Amazon S3 存储桶。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的将对象上传到存储桶。
3. 登录 AWS Management Console，然后通过以下网址打开 AWS Glue Studio 控制台：https://console.aws.amazon.com/gluestudio/。
4. 选择连接器，然后选择创建自定义连接器。
5. 对于连接器 S3 URL，选择浏览 S3，然后选择您上传到 Amazon S3 存储桶的 JDBC 驱动程序文件。
6. 为您的连接器输入描述性名称。例如，输入 **SQLServer**。
7. 对于连接器类型，请选择 JDBC。
8. 在类名称中，输入 JDBC 驱动程序的主类名称。对于 SQL Server，请输入 **com.microsoft.sqlserver.jdbc.SQLServerDriver**。
9. 对于 JDBC URL 库，请输入 JDBC 基本 URL。JDBC 基本 URL 的语法取决于源数据库引擎。对于 SQL Server，请使用以下格式：**jdbc:sqlserver://$*<host>*:$*<port>*;databaseName=$*<dbname>*;user=$*<username>*;password=$*<password>***。

   确保使用值替换 *<host>*、*<port>*、*<dbname>*、*<username>* 和 *<password>*。
10. 在 URL 参数分隔符中，输入分号（**;**）。
11. 选择 Create connector (创建连接器)。

在 AWS Secrets Manager 中存储数据库凭证

1. 登录 AWS Management Console ，然后通过以下网址打开 AWS Secrets Manager 控制台：https://console.aws.amazon.com/secretsmanager/。
2. 选择 Store a new secret (存储新密钥)。
3. 在 Choose secret type（选择密钥类型）页面上，执行以下操作：

   a. 对于密钥类型，请选择其他密钥类型。

   b. 对于键/值对，输入以下密钥：**host**、**port**、**dbname**、**username** 和 **password**。

   接下来，输入这些密钥的值。
4. 在配置密钥页面上，输入描述性密钥名称。例如，输入 **SQL_Server_secret**。

5. 选择 Next（下一步）。然后，在配置旋转页面再次选择下一步。
6. 在 Review (审核) 页上，审核您的密钥详细信息，然后选择 Store (存储)。

创建连接器连接

1. 登录 AWS Management Console，然后通过以下网址打开 AWS Glue Studio 控制台：https://console.aws.amazon.com/gluestudio/。
2. 选择要创建连接的连接器，然后选择创建连接。
3. 在创建连接页面上，输入连接的描述性名称。例如，输入 **SQL-Server-connection**。
4. 在 AWS 秘钥中，选择您在 AWS Secrets Manager 中创建的密钥。
5. 配置网络选项，然后选择创建连接。

现在，您可以使用自定义连接器创建 AWS Glue Studio 作业。有关更多信息，请参阅创建 AWS Glue Studio 作业。

将 SSIS 包添加到 AWS SCT 项目中

您可以将多个 SSIS 包添加到单个 AWS SCT 项目中。

将 SSIS 包添加到 AWS SCT 项目中

1. 在 AWS SCT 中创建新项目，或打开现有项目。有关更多信息，请参阅the section called “创建项目”。
2. 从菜单中选择添加源，然后选择 SQL Server Integration Services。
3. 在连接名称中，输入 SSIS 包的名称。AWS SCT 会在左侧面板的树中显示此名称。
4. 在 SSIS 包文件夹中，输入包含源 SSIS 包的文件夹的路径。
5. 从菜单中选择添加目标，然后选择 AWS Glue Studio。

   AWS SCT 使用 AWS 配置文件连接 AWS Glue Studio。有关更多信息，请参阅将 AWS 服务配置文件存储在 AWS SCT 中。
6. 创建映射规则，其中包括源 SSIS 包和 AWS Glue Studio 目标。有关更多信息，请参阅在 AWS SCT 中创建映射规则。
7. 在 AWS Glue Studio 控制台中创建 AWS Glue Studio 连接。有关更多信息，请参创建连接器连接。
8. 选择左侧树中的连接管理器，打开上下文（右键单击）菜单，然后选择配置连接。

AWS SCT 显示配置连接窗口。

9. 对于每个源 SSIS 连接，请选择一个 AWS Glue Studio 连接。

用 AWS SCT 将 SSIS 包转换为 AWS Glue Studio

接下来，了解如何使用 AWS SCT 将 SSIS 包转换为 AWS Glue Studio。

将 SSIS 包转换为 AWS Glue Studio

1. 将 SSIS 包添加到 AWS SCT 项目中。有关更多信息，请参阅将 SSIS 包添加到 AWS SCT 项目中。
2. 在左侧面板中，展开 ETL 和 SSIS 节点。
3. 选择包，打开上下文（右键单击）菜单，然后选择转换包。

   AWS SCT 将您选择的 SSIS 包转换为 JSON 文件。这些 JSON 对象表示有向无环图（DAG）中的节点。在右侧树的包 DAG 节点中找到转换后的文件。
4. 选择包 DAG，打开上下文（右键单击）菜单，然后选择保存到 Amazon S3。

   现在，您可以使用这些脚本在 AWS Glue Studio 中创建作业。

使用转换后的代码创建 AWS Glue Studio 作业

转换源 SSIS 包后，您可以使用转换后的 JSON 文件创建 AWS Glue Studio 作业。

创建 AWS Glue Studio 作业

1. 在右侧树中选择包 DAG，打开上下文（右键单击）菜单，然后选择配置 AWS Glue Studio 作业。
2. （可选）应用在 AWS Glue Studio 中模拟 SSIS 函数的扩展包。
3. 将打开配置 AWS Glue Studio 作业窗口。

   完成基本作业属性部分：

   - 名称：输入 AWS Glue Studio 作业名称。
   - 脚本文件名：输入作业脚本的名称。
   - 作业参数：添加参数并输入参数值。

选择 Next（下一步）。

4. 完成高级作业属性部分：

   - IAM 角色：选择用于 AWS Glue Studio 身份验证和访问数据存储的 IAM 角色。
   - 脚本文件 S3 路径：输入转换后的脚本的 Amazon S3 路径。
   - 临时目录：输入临时目录的 Amazon S3 路径以获得中间结果。AWS Glue Studio 使用此目录读取或写入 Amazon Redshift。
   - AWS SCT 会自动生成 Python 库的路径。你可以在生成的 python 库路径中查看此路径。您不能编辑这个自动生成的路径。要使用其他 Python 库，请在用户 python 库路径中输入路径。
   - 用户 python 库路径：输入其他用户 Python 库的路径。使用逗号分隔 Amazon S3 路径。
   - 从属 JAR 路径：输入从属 `*.jar` 文件的路径。使用逗号分隔 Amazon S3 路径。
   - 引用文件路径：输入脚本所需的其他文件（例如配置文件）的路径。使用逗号分隔 Amazon S3 路径。
   - 工作线程类型：选择 `G.1X` 或 `G.2X`。

     选择 `G.1X` 时，每个工作线程映射到 1 个 DPU（4 个 vCPU，16 GB 内存和 64 GB 磁盘）。

     选择 `G.2X` 时，每个工作线程映射到 2 个 DPU（8 个 vCPU，32 GB 内存和 128 GB 磁盘）。
   - 请求的每个工作线程数：输入作业运行时分配的工作线程数。
   - 最大并发：输入此作业允许的并发运行的最大数量。默认值是 1。达到此阈值时，AWS Glue 将返回一个错误。
   - 作业超时（分钟）：输入 ETL 作业的超时值，以防作业失控。批处理作业的默认值为 2880 分钟（48 小时）。当作业执行时间超过此限制时，作业运行状态更改为 `TIMEOUT`。
   - 延迟通知阈值（分钟）：输入 AWS SCT 发送延迟通知之前的阈值（以分钟为单位）。
   - 重试次数：输入在失败时 AWS Glue 自动重新启动作业的次数（0–10）。达到超时限制的作业不会重新启动。默认值为 0。

   选择 Finish (结束)。

   AWS SCT 配置您选择的 AWS Glue Studio 作业。

5. 在右侧树中的 ETL 作业下找到您配置的作业。选择您配置的作业，打开上下文（右键单击）菜单，然后选择创建 AWS Glue Studio 作业。
6. 选择应用状态，并确保作业的状态值为成功。

7. 打开 AWS Glue Studio 控制台，选择刷新，然后选择作业。然后，选择 Run（运行）。

使用 AWS SCT 创建 SSIS 包评估报告

ETL 迁移评估报告提供了有关将 SSIS 包转换为与 AWS Glue Studio 兼容的格式的信息。评估报告包括 SSIS 包各组件的操作项。这些操作项显示哪些组件 AWS SCT 无法自动转换。

创建 ETL 迁移评估报告

1. 展开左侧面板中 ETL 下的 SSIS 节点。
2. 选择包，打开上下文（右键单击）菜单，然后选择创建报告。
3. 查看摘要选项卡。此处 AWS SCT 显示 ETL 迁移评估报告中的执行摘要信息。它包括 SSIS 包所有组件的转换结果。
4. （可选）将 ETL 迁移评估报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件：
   - 要将 ETL 迁移评估报告另存为 PDF 文件，请选择右上角的保存为 PDF。

     PDF 文件包含脚本转换执行摘要、操作项和建议。
   - 要将 ETL 迁移评估报告另存为 CSV 文件，请选择右上角的保存为 CSV。

     AWS SCT 创建三个 CSV 文件。这些文件包含操作项、推荐的操作以及转换脚本所需的估计人工操作的复杂性。
5. 选择操作项选项卡。此选项卡包含需要手动转换为 AWS Glue Studio 的项目列表。如果从列表中选择一个操作项，AWS SCT 会突出显示您源 SSIS 包中该操作项适用的项。

AWS SCT 可以转换为 AWS Glue Studio 的 SSIS 组件

您可以使用 AWS SCT 将 SSIS 数据流组件和参数转换为 AWS Glue Studio。

支持的数据流组件包括：

- ADO NET 目的地
- ADO NET 源
- 聚合
- 字符映射
- 条件拆分

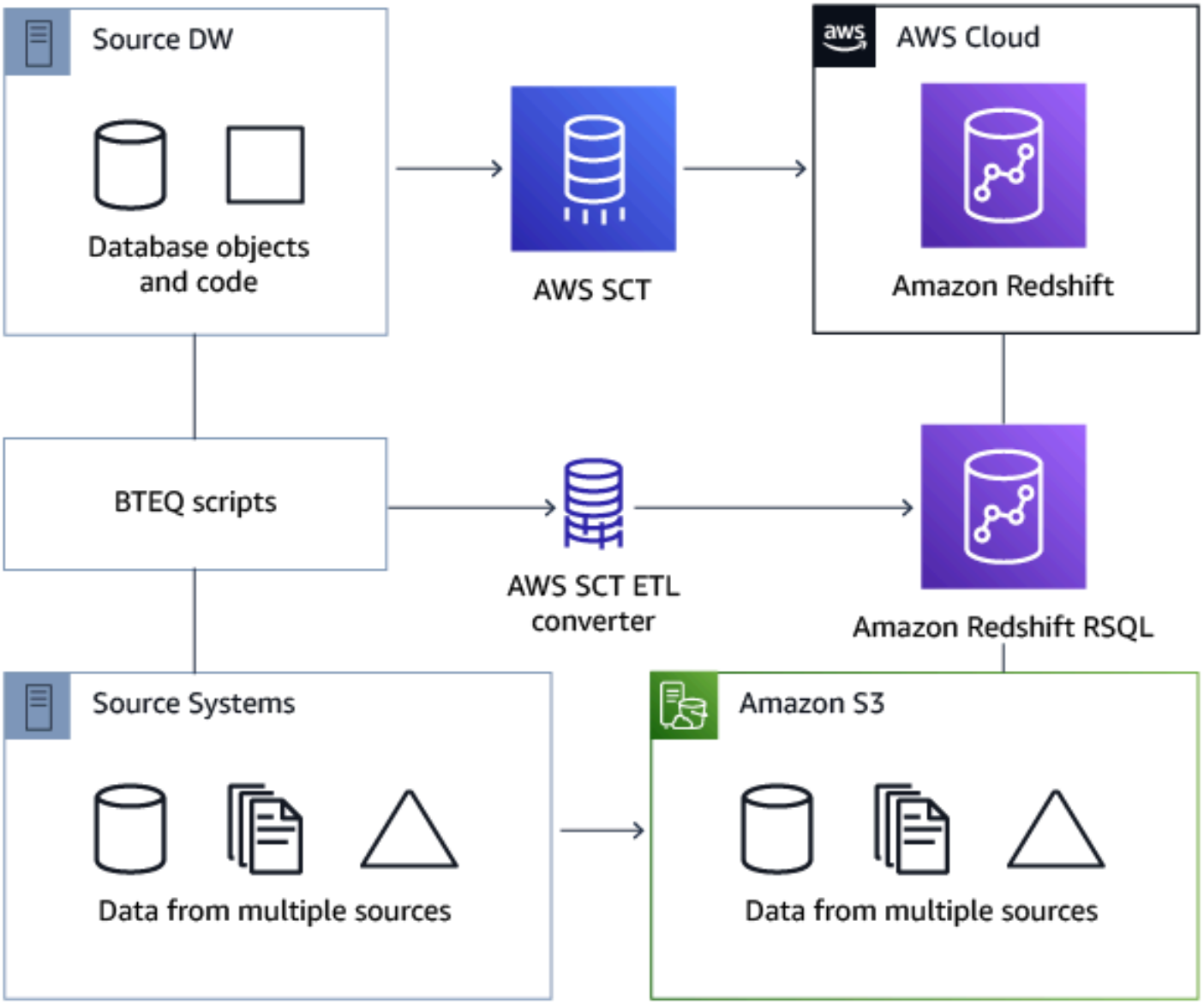
- 复制列
- 数据转换
- 派生列
- 查找
- 合并
- 合并联接
- 多播
- ODBCDestination
- ODBCSource
- OLEDBDestination
- OLEDBSource
- 行数
- 排序
- SQL Server 目的地
- UNION ALL

AWS SCT 可以将更多 SSIS 组件转换为 AWS Glue。有关更多信息，请参阅AWS SCT 可以转换为 AWS Glue 的 SSIS 组件。

使用 AWS SCT 将 Teradata BTEQ 脚本转换为 Amazon Redshift RSQL

您可以使用 AWS Schema Conversion Tool（AWS SCT）将 Teradata Basic Teradata Query（BTEQ）脚本转换为 Amazon Redshift RSQL。

以下架构图显示了数据库迁移项目，其中包括将提取、转换和加载（ETL）脚本转换为 Amazon Redshift RSQL。



主题

- 将 BTEQ 脚本添加到 AWS SCT 项目中
- 使用 AWS SCT 在 BTEQ 脚本中配置替代变量
- 使用 AWS SCT 将 Teradata BTEQ 脚本转换为 Amazon Redshift RSQL
- 使用 AWS SCT 管理 BTEQ 脚本
- 使用 AWS SCT 创建 BTEQ 脚本转换评估报告
- 使用 AWS SCT 编辑和保存转换后的 BTEQ 脚本

将 BTEQ 脚本添加到 AWS SCT 项目中

您可以将多个脚本添加到单个 AWS SCT 项目。

将 BTEQ 脚本添加到 AWS SCT 项目中

1. 在 AWS SCT 中创建新项目，或打开现有项目。有关更多信息，请参阅the section called “创建项目”。
2. 从菜单中选择添加源，然后选择 Teradata 将源数据库添加到项目中。有关更多信息，请参阅将 Teradata 用作源。
3. 从菜单中选择添加目标，将目标 Amazon Redshift 数据库添加到 AWS SCT 项目中。

   您可以使用虚拟的 Amazon Redshift 目标数据库平台。有关更多信息，请参阅使用虚拟目标。
4. 创建新的映射规则，其中包括源 Teradata 数据库和 Amazon Redshift 目标。有关更多信息，请参阅添加新映射规则。
5. 在视图菜单上，选择主视图。
6. 在左侧面板中，展开脚本节点。
7. 选择 BTEQ 脚本，打开上下文（右键单击）菜单，然后选择加载脚本。
8. 输入 Teradata BTEQ 脚本的源代码位置，然后选择选择文件夹。

   AWS SCT 显示加载脚本窗口。
9. 请执行下列操作之一：

   a. 如果您的 Teradata BTEQ 脚本不包含替代变量，请选择无替代变量，然后选择确定将脚本添加到 AWS SCT 项目中。

   b. 如果 Teradata BTEQ 脚本包含替代变量，请配置替代变量。有关更多信息，请参阅在 BTEQ 脚本中配置替代变量。

使用 AWS SCT 在 BTEQ 脚本中配置替代变量

Teradata BTEQ 脚本可以包含替代变量。例如，您可以使用一个带有替代变量的 BTEQ 脚本在多个数据库环境中运行同一组命令。您可以使用 AWS SCT 在 BTEQ 脚本中配置替代变量。

在运行带有替代变量的 BTEQ 脚本之前，请确保为所有变量分配值。为此，您可以使用其他工具或应用程序，例如 Bash 脚本、UC4（Automic）等。AWS SCT 只有在替代变量赋值后才能解析和转换替代变量。

在 BTEQ 脚本中配置替代变量

1. 将 BTEQ 脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 BTEQ 脚本添加到 AWS SCT 项目中。

   添加脚本时，选择使用替代变量。

2. 在定义变量格式中，输入与脚本中所有替代变量相匹配的正则表达式。

   例如，如果替代变量的名称以 ${ 开头且以 } 结尾，则使用 \$\{\w+\} 正则表达式。要匹配以美元符号或百分号开头的替代变量，请使用 \$\w+|\%\w+ 正则表达式。

   AWS SCT 中的正则表达式符合 Java 正则表达式语法。有关更多信息，请参阅 Java 文档中的 java.util.regex 类模式。

3. 选择确定，将脚本加载到 AWS SCT 项目中，然后选择确定关闭加载脚本窗口。
4. 选择变量可查看所有发现的替代变量及其值。
5. 在值中，输入替代变量的值。

使用 AWS SCT 将 Teradata BTEQ 脚本转换为 Amazon Redshift RSQL

接下来，了解如何使用 AWS SCT 将 BTEQ ETL 脚本转换为 Amazon Redshift RSQL。

将 Teradata BTEQ 脚本转换为 Amazon Redshift RSQL

1. 将 BTEQ 脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 BTEQ 脚本添加到 AWS SCT 项目中。
2. 配置替代变量。有关更多信息，请参阅在 BTEQ 脚本中配置替代变量。
3. 在左侧面板中，展开脚本节点。
4. 请执行下列操作之一：

   - 要转换单个 BTEQ 脚本，请展开 BTEQ 脚本节点，选择要转换的脚本，然后从上下文（右键单击）菜单中选择转换为 RSQL。
   - 要转换多个脚本，请确保选择所有要转换的脚本。然后选择 BTEQ 脚本，打开上下文（右键单击）菜单，然后在转换脚本下选择转换为 RSQL。

   AWS SCT 将您选定的所有 Teradata BTEQ 脚本转换为与 Amazon Redshift RSQL 兼容的格式。在目标数据库面板的脚本节点中找到转换后的脚本。

5. 编辑转换后的 Amazon Redshift RSQL 脚本或将其保存。有关更多信息，请参阅编辑和保存转换后的 BTEQ 脚本。

使用 AWS SCT 管理 BTEQ 脚本

您可以添加多个 BTEQ 脚本或从 AWS SCT 项目中移除一个 BTEQ 脚本。

将其他 BTEQ 脚本添加到 AWS SCT 项目中

1. 在左侧面板中，展开脚本节点。
2. 选择 BTEQ 脚本节点，然后打开上下文（右键单击）菜单。
3. 选择加载脚本。
4. 输入添加新 BTEQ 脚本和配置替代变量所需的信息。有关更多信息，请参阅 将 BTEQ 脚本添加到 AWS SCT 项目中 和 在 BTEQ 脚本中配置替代变量：

从 AWS SCT 项目中删除 BTEQ 脚本

1. 展开左侧面板中脚本下的 BTEQ 脚本节点。
2. 选择要删除的脚本，然后打开上下文 (右键单击) 菜单。
3. 选择删除脚本。

使用 AWS SCT 创建 BTEQ 脚本转换评估报告

BTEQ 脚本转换评估报告提供了有关将 BTEQ 脚本中的 BTEQ 命令和 SQL 语句转换为与 Amazon Redshift RSQL 兼容的格式的信息。评估报告包括 AWS SCT 无法转换的 BTEQ 命令和 SQL 语句的操作项。

创建 BTEQ 脚本转换评估报告

1. 展开左侧面板中脚本下的 BTEQ 脚本节点。
2. 选择要转换的脚本，然后打开上下文 (右键单击) 菜单。
3. 在创建报告下选择转换为 RSQL。
4. 查看摘要选项卡。摘要选项卡显示了来自 BTEQ 脚本评估报告的执行摘要信息。它包括 BTEQ 脚本中的所有 BTEQ 命令以及 SQL 语句的转换结果。
5. （可选）将 BTEQ 脚本转换评估报告的本地副本另存为 PDF 文件或逗号分隔值 (CSV) 文件：

- 要将 BTEQ 脚本转换评估报告另存为 PDF 文件，请选择右上角的保存为 PDF。

  PDF 文件包含脚本转换执行摘要、操作项和建议。
- 要将 BTEQ 脚本转换评估报告另存为 CSV 文件，请选择右上角的保存为 CSV。

  CSV 文件包含操作项、推荐的操作以及转换脚本所需的估计人工操作的复杂度。

6. 选择操作项选项卡。此选项卡包含需要手动转换为 Amazon Redshift RSQL 的项目列表。从列表中选择一个操作项时，AWS SCT 会突出显示源 BTEQ 脚本中该操作项适用的项。

使用 AWS SCT 编辑和保存转换后的 BTEQ 脚本

您可以在 AWS SCT 项目的下部面板中编辑转换后的脚本。AWS SCT 将编辑后的脚本作为项目的一部分存储。

保存转换后的脚本

1. 展开目标数据库面板中脚本下的 RSQL 脚本节点。
2. 选择转换后的脚本，打开上下文（右键单击）菜单，然后选择保存脚本。
3. 输入用于保存转换后的脚本的文件夹路径，然后选择保存。

   AWS SCT 将转换后的脚本保存到文件中并打开此文件。

使用 AWS SCT 将带有嵌入式 Teradata BTEQ 命令的 Shell 脚本转换为 Amazon Redshift RSQL

您可以使用 AWS Schema Conversion Tool（AWS SCT）将带有嵌入式 Teradata Basic Teradata Query（BTEQ）命令的 Shell 脚本转换为带有嵌入式 Amazon Redshift RSQL 命令的 Shell 脚本。

AWS SCT 从 Shell 脚本中提取 Teradata BTEQ 命令并将其转换为与 Amazon Redshift 兼容的格式。将 Teradata 数据库迁移到 Amazon Redshift 后，您可以使用这些转换后的脚本管理新的 Amazon Redshift 数据库。

你也可以使用 AWS SCT 将带有 Teradata BTEQ ETL 脚本的文件转换为 Amazon Redshift RSQL。有关更多信息，请参阅使用 AWS SCT 将 Teradata BTEQ 脚本转换为 Amazon Redshift RSQL。

主题

将带有嵌入式 Teradata BTEQ 命令的 Shell 脚本添加到 AWS SCT 项目中

您可以将多个脚本添加到单个 AWS SCT 项目。

将 Shell 脚本添加到 AWS SCT 项目中

1. 在 AWS SCT 中创建新项目，或打开现有项目。有关更多信息，请参阅the section called “创建项目”。
2. 从菜单中选择添加源，然后选择 Teradata 将源数据库添加到项目中。有关更多信息，请参阅将 Teradata 用作源。
3. 从菜单中选择添加目标，然后将目标 Amazon Redshift 数据库添加到 AWS SCT 项目中。

   您可以使用虚拟的 Amazon Redshift 目标数据库平台。有关更多信息，请参阅使用虚拟目标。
4. 创建新的映射规则，其中包括源 Teradata 数据库和 Amazon Redshift 目标。有关更多信息，请参阅添加新映射规则。
5. 在视图菜单上，选择主视图。
6. 在左侧面板中，展开脚本节点。
7. 选择 Shell，打开上下文（右键单击）菜单，然后选择加载脚本。
8. 输入包含嵌入式 Teradata BTEQ 命令的源 Shell 脚本的位置，然后选择选择文件夹。

   AWS SCT 显示加载脚本窗口。
9. 请执行下列操作之一：

   - 如果 Shell 脚本不包含替代变量，请选择无替代变量，然后选择确定将脚本添加到 AWS SCT 项目中。
   - 如果 Shell 脚本包含替代变量，请配置替代变量。有关更多信息，请参阅在 Shell 脚本中配置替代变量。

使用 AWS SCT 在带有嵌入式 Teradata BTEQ 命令的 Shell 脚本中配置替代变量

Shell 脚本可以包含替代变量。例如，您可以使用带有替代变量的单个脚本管理不同环境中的数据库。您可以使用 AWS SCT 在 Shell 脚本中配置替代变量。

在使用 Shell 脚本中的替代变量运行 BTEQ 命令之前，请确保为此 Shell 脚本中的所有变量分配值。AWS SCT 只有在替代变量赋值后才能解析和转换替代变量。

在 Shell 脚本中配置替代变量

1. 将源 Shell 脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 Shell 脚本添加到 AWS SCT 项目中。

   添加脚本时，选择使用替代变量。
2. 在定义变量格式中，输入与脚本中所有替代变量相匹配的正则表达式。

   例如，如果替代变量的名称以 ${ 开头且以 } 结尾，则使用 \$\{\w+\} 正则表达式。要匹配以美元符号或百分号开头的替代变量，请使用 \$\w+|\%\w+ 正则表达式。

   AWS SCT 中的正则表达式符合 Java 正则表达式语法。有关更多信息，请参阅 Java 文档中的 java.util.regex 类模式。
3. 选择确定将脚本加载到 AWS SCT 项目中，然后选择确定关闭加载脚本窗口。
4. 选择变量可查看所有发现的替代变量及其值。
5. 在值中，输入替代变量的值。

使用 AWS SCT 转换带嵌入式 Teradata BTEQ 命令的 Shell 脚本

接下来，了解如何使用 AWS SCT 将带有嵌入式 Teradata BTEQ 命令的 Shell 脚本转换为带有嵌入式 Amazon Redshift RSQL 命令的 Shell 脚本。

转换 Shell 脚本

1. 将 Shell 脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 Shell 脚本添加到 AWS SCT 项目中。
2. 配置替代变量。有关更多信息，请参阅在 Shell 脚本中配置替代变量。
3. 在左侧面板中，展开脚本节点。

4. 请执行下列操作之一：

   - 要转换来自单个 Shell 脚本的 BTEQ 命令，请展开 Shell 节点，选择要转换的脚本，然后从上下文（右键单击）菜单中选择转换脚本。
   - 要转换多个脚本，请确保选择所有要转换的脚本。然后选择 Shell，打开上下文（右键单击）菜单，再选择转换脚本。

5. 选择确定。

   AWS SCT 将所选 Shell 脚本中的 BTEQ 命令转换为与 Amazon Redshift RSQL 兼容的格式。在目标数据库面板的脚本节点中找到转换后的脚本。

6. 编辑转换后的 Amazon Redshift RSQL 脚本或将其保存。有关更多信息，请参阅编辑和保存转换后的 Shell 脚本。

使用 AWS SCT 管理带有嵌入式 Teradata BTEQ 命令的 Shell 脚本

您可以添加多个 Shell 脚本或从 AWS SCT 项目中移除一个 Shell 脚本。

将新的 Shell 脚本添加到 AWS SCT 项目中

1. 在左侧面板中，展开脚本节点。
2. 选择 Shell 节点，然后打开上下文 (右键单击) 菜单。
3. 选择加载脚本。
4. 输入添加新 Shell 脚本和配置替代变量所需的信息。有关更多信息，请参阅 将 Shell 脚本添加到 AWS SCT 项目中 和 在 Shell 脚本中配置替代变量：

从 AWS SCT 项目中删除 Shell 脚本

1. 展开左侧面板中脚本下的 Shell 节点。
2. 选择要删除的脚本，然后打开上下文 (右键单击) 菜单。
3. 选择删除脚本。

使用 AWS SCT 创建 Shell 脚本转换评估报告

Shell 脚本转换评估报告提供了有关转换 BTEQ 命令和 SQL 语句的信息。将源脚本转换为与 Amazon Redshift RSQL 兼容的格式。评估报告包括 AWS SCT 无法转换的 BTEQ 命令和 SQL 语句的操作项。

创建 Shell 脚本转换评估报告

1. 展开左侧面板中脚本下的 Shell 节点。
2. 选择要转换的脚本，打开上下文（右键单击）菜单，然后选择创建报告。
3. 查看摘要选项卡。摘要选项卡显示 Shell 脚本评估报告的执行摘要信息。它包括源脚本中所有 BTEQ 命令和 SQL 语句的转换结果。
4. （可选）将 Shell 脚本转换评估报告的本地副本另存为 PDF 文件或逗号分隔值 (CSV) 文件：
   - 要将 Shell 脚本转换评估报告另存为 PDF 文件，请选择右上角的保存为 PDF。

     PDF 文件包含脚本转换执行摘要、操作项和建议。
   - 要将 Shell 脚本转换评估报告另存为 CSV 文件，请选择右上角的保存为 CSV。

     CSV 文件包含操作项、推荐的操作以及转换脚本所需的估计人工操作的复杂度。
5. 选择操作项选项卡。此选项卡包含需要手动转换为 Amazon Redshift RSQL 的项目列表。从列表中选择一个操作项时，AWS SCT 会突出显示源 Shell 脚本中该操作项适用的项。

使用 AWS SCT 编辑和保存转换后的 Shell 脚本

您可以在 AWS SCT 项目的下部面板中编辑转换后的脚本。AWS SCT 将编辑后的脚本作为项目的一部分存储。

保存转换后的脚本

1. 展开目标数据库面板中脚本下的 RSQL 脚本节点。
2. 选择转换后的脚本，打开上下文（右键单击）菜单，然后选择保存脚本。
3. 输入用于保存转换后的脚本的文件夹路径，然后选择保存。

   AWS SCT 将转换后的脚本保存到文件中并打开此文件。

使用 AWS SCT 将 Teradata FastExport 作业脚本转换为 Amazon Redshift RSQL

您可以使用 AWS Schema Conversion Tool（AWS SCT）将 Teradata FastExport 作业脚本转换为 Amazon Redshift RSQL。

FastExport 作业脚本是一组 FastExport 命令和 SQL 语句，用于从 Teradata 数据库中选择和导出数据。AWS SCT 将 FastExport 命令和 SQL 语句转换为与 Amazon Redshift RSQL 兼容的格式。将 Teradata 数据库迁移到 Amazon Redshift 后，您可以使用这些转换后的脚本从 Amazon Redshift 数据库导出数据。

主题

将 FastExport 作业脚本添加到 AWS SCT 项目中

您可以将多个脚本添加到单个 AWS SCT 项目。

将 FastExport 作业脚本添加到 AWS SCT 项目中

1. 在 AWS SCT 中创建新项目，或打开现有项目。有关更多信息，请参阅the section called “创建项目”。
2. 从菜单中选择添加源，然后选择 Teradata 将源数据库添加到项目中。有关更多信息，请参阅将 Teradata 用作源。
3. 从菜单中选择添加目标，然后将目标 Amazon Redshift 数据库添加到 AWS SCT 项目中。

   您可以使用虚拟的 Amazon Redshift 目标数据库平台。有关更多信息，请参阅使用虚拟目标。
4. 创建新的映射规则，其中包括源 Teradata 数据库和 Amazon Redshift 目标。有关更多信息，请参阅添加新映射规则。
5. 在视图菜单上，选择主视图。
6. 在左侧面板中，展开脚本节点。
7. 选择 FastExport，打开上下文（右键单击）菜单，然后选择加载脚本。
8. 输入 Teradata FastExport 作业脚本的源代码位置，然后选择选择文件夹。

   AWS SCT 显示加载脚本窗口。
9. 请执行下列操作之一：

- 如果 Teradata FastExport 作业脚本不包含替代变量，请选择无替代变量，然后选择确定将脚本添加到 AWS SCT 项目中。
- 如果 Teradata FastExport 作业脚本包含替代变量，请配置替代变量。有关更多信息，请参阅在 FastExport 作业脚本中配置替代变量。

在 Teradata FastExport 作业脚本中使用 AWS SCT 配置替代变量

Teradata FastExport 作业脚本可以包含替代变量。例如，您可以使用带有替代变量的单个脚本从多个数据库导出数据。您可以使用 AWS SCT 在 Teradata 脚本中配置替代变量。

运行带有替代变量的 FastExport 作业脚本之前，请确保为所有变量分配值。为此，您可以使用其他工具或应用程序，例如 Bash 脚本、UC4 (Automic) 等。AWS SCT 只有在替代变量赋值后才能解析和转换替代变量。

在 FastExport 作业脚本中配置替代变量

1. 将源 Teradata FastExport 作业脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 BTEQ 脚本添加到 AWS SCT 项目中。

   添加脚本时，选择使用替代变量。
2. 在定义变量格式中，输入与脚本中所有替代变量相匹配的正则表达式。

   例如，如果替代变量的名称以 `${` 开头且以 `}` 结尾，则使用 `\$\{\w+\}` 正则表达式。要匹配以美元符号或百分号开头的替代变量，请使用 `\$\w+|\%\w+` 正则表达式。

   AWS SCT 中的正则表达式符合 Java 正则表达式语法。有关更多信息，请参阅 Java 文档中的 java.util.regex 类模式。
3. 选择确定，将脚本加载到 AWS SCT 项目中，然后选择确定关闭加载脚本窗口。
4. 在左侧面板中，展开脚本节点。选择 FastExport，然后选择包含脚本的文件夹。打开上下文（右键单击）菜单，然后选择替代变量下的导出变量。
5. 导出一个脚本的替代变量。展开包含脚本的文件夹，选择脚本，打开上下文（右键单击）菜单，然后选择替代变量下的导出变量。
6. 输入逗号分隔值（CSV）文件名以保存替代变量，然后选择保存。
7. 打开此 CSV 文件并填写替代变量的值。

根据操作系统的不同，AWS SCT 使用不同格式的 CSV 文件。文件中的值可以用引号括起来，也可以不用引号括起来。确保替代变量的值与文件中其他值格式相同。AWS SCT 无法导入具有不同格式值的 CSV 文件。

8. 保存 CSV 文件。
9. 在左侧面板中，展开脚本节点。选择 FastExport，然后选择脚本。打开上下文（右键单击）菜单，然后选择替代变量下的导入变量。
10. 选择 CSV 文件，然后选择打开。
11. 选择变量可查看所有发现的替代变量及其值。

使用 AWS SCT 转换 Teradata FastExport 作业脚本

接下来，了解如何使用 AWS SCT 将 Teradata FastExport 作业转换为 Amazon Redshift RSQL。

将 Teradata FastExport 作业脚本转换为 Amazon Redshift RSQL

1. 将 FastExport 作业脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 FastExport 作业脚本添加到 AWS SCT 项目中。
2. 配置替代变量。有关更多信息，请参阅在 FastExport 作业脚本中配置替代变量。
3. 在左侧面板中，展开脚本节点。
4. 请执行下列操作之一：

   - 要转换单个 FastExport 作业脚本，请展开 FastExport 节点，选择要转换的脚本，然后从上下文（右键单击）菜单中选择转换脚本。
   - 要转换多个脚本，请确保选择所有要转换的脚本。然后选择 FastExport，打开上下文（右键单击）菜单，然后选择转换脚本。

   AWS SCT 将所有选定的 Teradata FastExport 作业脚本转换为与 Amazon Redshift RSQL 兼容的格式。在目标数据库面板的脚本节点中找到转换后的脚本。
5. 编辑转换后的 Amazon Redshift RSQL 脚本或将其保存。有关更多信息，请参阅编辑和保存转换后的 FastExport 作业脚本。

使用 AWS SCT 管理 Teradata FastExport 作业脚本

您可以添加多个 Teradata FastExport 作业脚本或从 AWS SCT 项目中移除一个 FastExport 作业脚本。

将新的 FastExport 作业脚本添加到 AWS SCT 项目中

1. 在左侧面板中，展开脚本节点。
2. 选择 FastExport 节点，然后打开上下文菜单 (右键单击)。
3. 选择加载脚本。
4. 输入添加新的 FastExport 作业脚本和配置替代变量所需的信息。有关更多信息，请参阅 将 FastExport 作业脚本添加到 AWS SCT 项目中 和 在 FastExport 作业脚本中配置替代变量：

从 AWS SCT 项目中移除 FastExport 作业脚本

1. 展开左侧面板脚本下的 FastExport 节点。
2. 选择要删除的脚本，然后打开上下文 (右键单击) 菜单。
3. 选择删除脚本。

使用 AWS SCT 创建 Teradata FastExport 作业脚本转换评估报告

FastExport 作业脚本转换评估报告提供了有关将 FastExport 脚本的 FastExport 命令和 SQL 语句转换为与 Amazon Redshift RSQL 兼容的格式的信息。评估报告包括 AWS SCT 无法转换的 FastExport 命令和 SQL 语句的操作项。

创建 Teradata FastExport 作业脚本转换评估报告

1. 展开左侧面板脚本下的 FastExport 节点。
2. 选择要转换的脚本，打开上下文（右键单击）菜单，然后选择创建报告。
3. 查看摘要选项卡。摘要选项卡显示 FastExport 作业脚本评估报告的执行摘要信息。它包括源脚本中所有 FastExport 命令和 SQL 语句的转换结果。
4. 您可以将 FastExport 作业脚本转换评估报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。

    a. 要将 FastExport 作业脚本转换评估报告另存为 PDF 文件，请选择右上角的保存为 PDF。

        PDF 文件包含脚本转换执行摘要、操作项和建议。

b. 要将 FastExport 作业脚本转换评估报告另存为 CSV 文件，请选择右上角的保存为 CSV。

CSV 文件包含操作项、推荐的操作以及转换脚本所需的估计人工操作的复杂度。

5. 选择操作项选项卡。此选项卡包含需要手动转换为 Amazon Redshift RSQL 的项目列表。从列表中选择一个操作项时，AWS SCT 会突出显示源 FastExport 作业脚本中该操作项适用的项。

使用 AWS SCT 编辑和保存转换后的 Teradata FastExport 作业脚本

您可以在 AWS SCT 项目的下部面板中编辑转换后的脚本。AWS SCT 将编辑后的脚本作为项目的一部分存储。

保存转换后的脚本

1. 展开目标数据库面板中脚本下的 RSQL 脚本节点。
2. 选择转换后的脚本，打开上下文（右键单击）菜单，然后选择保存脚本。
3. 输入用于保存转换后的脚本的文件夹路径，然后选择保存。

AWS SCT 将转换后的脚本保存到文件中并打开此文件。

使用 AWS SCT 将 Teradata FastLoad 作业脚本转换为 Amazon Redshift RSQL

您可以使用 AWS Schema Conversion Tool（AWS SCT）将 Teradata FastLoad 作业脚本转换为 Amazon Redshift RSQL。

Teradata FastLoad 脚本是一组命令，它们使用多个会话将数据加载到 Teradata 数据库的空表中。Teradata FastLoad 处理一系列 Teradata FastLoad 命令和 SQL 语句。Teradata FastLoad 命令为数据传输提供会话控制和数据处理。SQL 语句创建、维护和删除表。

AWS SCT 将 Teradata FastLoad 命令和 SQL 语句转换为与 Amazon Redshift RSQL 兼容的格式。将 Teradata 数据库迁移到 Amazon Redshift 后，您可以使用这些转换后的脚本将数据加载到 Amazon Redshift 数据库。

主题

- 将 FastLoad 作业脚本添加到 AWS SCT 项目中
- 在 Teradata FastLoad 作业脚本中使用 AWS SCT 配置替代变量
- 使用 AWS SCT 转换 Teradata FastLoad 作业脚本

- 使用 AWS SCT 管理 Teradata FastLoad 作业脚本
- 使用 AWS SCT 创建 Teradata FastLoad 作业脚本转换评估报告
- 使用 AWS SCT 编辑和保存转换后的 Teradata FastLoad 作业脚本

将 FastLoad 作业脚本添加到 AWS SCT 项目中

您可以将多个脚本添加到单个 AWS SCT 项目。

将 FastLoad 作业脚本添加到 AWS SCT 项目中

1. 在 AWS SCT 中创建新项目，或打开现有项目。有关更多信息，请参阅the section called “创建项目”。
2. 从菜单中选择添加源，然后选择 Teradata 将源数据库添加到项目中。有关更多信息，请参阅将 Teradata 用作源。
3. 从菜单中选择添加目标，然后将目标 Amazon Redshift 数据库添加到 AWS SCT 项目中。

   您可以使用虚拟的 Amazon Redshift 目标数据库平台。有关更多信息，请参阅使用虚拟目标。
4. 创建新的映射规则，其中包括源 Teradata 数据库和 Amazon Redshift 目标。有关更多信息，请参阅添加新映射规则。
5. 在视图菜单上，选择主视图。
6. 在左侧面板中，展开脚本节点。
7. 选择 FastLoad，打开上下文（右键单击）菜单，然后选择加载脚本。
8. 输入源 Teradata FastLoad 作业脚本的位置，然后选择选择文件夹。

   AWS SCT 显示加载脚本窗口。
9. 请执行下列操作之一：

   - 如果 Teradata FastLoad 作业脚本不包含替代变量，请选择无替代变量，然后选择确定将脚本添加到 AWS SCT 项目中。
   - 如果 Teradata FastLoad 作业脚本包含替代变量，请配置替代变量。有关更多信息，请参阅在 FastLoad 作业脚本中配置替代变量。

在 Teradata FastLoad 作业脚本中使用 AWS SCT 配置替代变量

Teradata FastLoad 作业脚本可能包含替代变量。例如，您可以使用带有替代变量的单个脚本将数据加载到不同的数据库。

运行带有替代变量的 FastLoad 作业脚本之前，请确保为所有变量分配值。为此，您可以使用其他工具或应用程序，例如 Bash 脚本、UC4（Automic）等。

AWS SCT 只有在替代变量赋值后才能解析和转换替代变量。在开始转换源 Teradata FastLoad 作业脚本之前，请确保为所有替代变量分配值。您可以使用 AWS SCT 在 Teradata 脚本中配置替代变量。

在 FastLoad 作业脚本中配置替代变量

1. 将源 Teradata FastLoad 作业脚本添加到 AWS SCT 项目中时，请选择使用替代变量。有关添加这些脚本的更多信息，请参阅 将 FastLoad 作业脚本添加到 AWS SCT 项目中。
2. 在定义变量格式中，输入与脚本中所有替代变量相匹配的正则表达式。

   例如，如果替代变量的名称以 ${ 开头且以 } 结尾，则使用 \$\{\w+\} 正则表达式。要匹配以美元符号或百分号开头的替代变量，请使用 \$\w+|\%\w+ 正则表达式。

   AWS SCT 中的正则表达式符合 Java 正则表达式语法。有关更多信息，请参阅 Java 文档中的 java.util.regex 类模式。
3. 选择确定，将脚本加载到 AWS SCT 项目中，然后选择确定关闭加载脚本窗口。
4. 在左侧面板中，展开脚本节点。选择 FastLoad，然后选择包含脚本的文件夹。打开上下文（右键单击）菜单，然后选择替代变量下的导出变量。

   此外，您可以导出一个脚本的替代变量。展开包含脚本的文件夹，选择脚本，打开上下文（右键单击）菜单，然后选择替代变量下的导出变量。
5. 输入逗号分隔值 (CSV) 文件名以保存替代变量，然后选择保存。
6. 打开此 CSV 文件并填写替代变量的值。

   根据操作系统的不同，AWS SCT 使用不同格式的 CSV 文件。文件中的值可以用引号括起来，也可以不用引号括起来。确保替代变量的值与文件中其他值格式相同。AWS SCT 无法导入具有不同格式值的 CSV 文件。
7. 保存 CSV 文件。
8. 在左侧面板中，展开脚本节点。选择 FastLoad，然后选择脚本。打开上下文（右键单击）菜单，然后选择替代变量下的导入变量。
9. 选择 CSV 文件，然后选择打开。
10. 选择变量可查看所有发现的替代变量及其值。

使用 AWS SCT 转换 Teradata FastLoad 作业脚本

接下来，了解如何使用 AWS SCT 将 Teradata FastLoad 作业转换为 Amazon Redshift RSQL。

将 Teradata FastLoad 作业脚本转换为 Amazon Redshift RSQL

1. 将 FastLoad 作业脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 FastLoad 作业脚本添加到 AWS SCT 项目中。
2. 配置替代变量。有关更多信息，请参阅在 FastLoad 作业脚本中配置替代变量。
3. 在左侧面板中，展开脚本节点。
4. 请执行下列操作之一：

   - 要转换单个 FastLoad 作业脚本，请展开 FastLoad 节点，选择要转换的脚本，然后从上下文（右键单击）菜单中选择转换脚本。
   - 要转换多个脚本，请确保选择所有要转换的脚本。选择 FastLoad，打开上下文（右键单击）菜单，然后选择转换脚本。然后，执行以下操作之一：
     - 如果您将源数据文件存储在 Amazon S3 上，请选择源数据文件位置的 S3 对象路径。

       为源数据文件输入 Amazon S3 存储桶文件夹和清单文件 Amazon S3 存储桶的值。
     - 如果您未将源数据文件存储在 Amazon S3 上，请选择源数据文件位置中的主机地址。

       为源数据文件输入主机的 URL 或 IP 地址、主机用户登录名和清单文件 Amazon S3 存储桶的值。
5. 选择确定。

   AWS SCT 将所有选定的 Teradata FastLoad 作业脚本转换为与 Amazon Redshift RSQL 兼容的格式。在目标数据库面板的脚本节点中找到转换后的脚本。
6. 编辑转换后的 Amazon Redshift RSQL 脚本或将其保存。有关更多信息，请参阅编辑和保存转换后的 FastLoad 作业脚本。

使用 AWS SCT 管理 Teradata FastLoad 作业脚本

您可以添加多个 Teradata FastLoad 作业脚本或从 AWS SCT 项目中移除一个 FastLoad 作业脚本。

将新的 FastLoad 作业脚本添加到 AWS SCT 项目中

1. 在左侧面板中，展开脚本节点。

2. 选择 FastLoad 节点，然后打开上下文 (右键单击) 菜单。
3. 选择加载脚本。
4. 输入添加新的 FastLoad 作业脚本和配置替代变量所需的信息。有关更多信息，请参阅 将 FastLoad 作业脚本添加到 AWS SCT 项目中 和 在 FastLoad 作业脚本中配置替代变量：

从 AWS SCT 项目中移除 FastLoad 作业脚本

1. 展开左侧面板脚本下的 FastLoad 节点。
2. 选择要删除的脚本，然后打开上下文 (右键单击) 菜单。
3. 选择删除脚本。

使用 AWS SCT 创建 Teradata FastLoad 作业脚本转换评估报告

FastLoad 作业脚本转换评估报告提供了有关转换 FastLoad 命令和 SQL 语句的信息。将源脚本转换为与 Amazon Redshift RSQL 兼容的格式。评估报告包括 AWS SCT 无法转换的 FastLoad 命令和 SQL 语句的操作项。

创建 Teradata FastLoad 作业脚本转换评估报告

1. 展开左侧面板脚本下的 FastLoad 节点。
2. 选择要转换的脚本，打开上下文（右键单击）菜单，然后选择创建报告。
3. 查看摘要选项卡。

   摘要选项卡显示 FastLoad 作业脚本评估报告的执行摘要信息。它包括源脚本中所有 FastLoad 命令和 SQL 语句的转换结果。
4. （可选）将 FastLoad 作业脚本转换评估报告的本地副本另存为 PDF 文件或逗号分隔值 (CSV) 文件：

   - 要将 FastLoad 作业脚本转换评估报告另存为 PDF 文件，请选择右上角的保存为 PDF。

     PDF 文件包含脚本转换的执行摘要、操作项和建议。
   - 要将 FastLoad 作业脚本转换评估报告另存为 CSV 文件，请选择右上角的保存为 CSV。

     CSV 文件包含操作项、推荐的操作以及转换脚本所需的估计人工操作的复杂度。
5. 选择操作项选项卡。此选项卡包含需要手动转换为 Amazon Redshift RSQL 的项目列表。从列表中选择一个操作项时，AWS SCT 会突出显示源 FastLoad 作业脚本中该操作项适用的项。

使用 AWS SCT 编辑和保存转换后的 Teradata FastLoad 作业脚本

您可以在 AWS SCT 项目的下部面板中编辑转换后的脚本。AWS SCT 将编辑后的脚本作为项目的一部分存储。

保存转换后的脚本

1. 展开目标数据库面板中脚本下的 RSQL 脚本节点。
2. 选择转换后的脚本，打开上下文（右键单击）菜单，然后选择保存脚本。
3. 输入用于保存转换后的脚本的文件夹路径，然后选择保存。

   AWS SCT 将转换后的脚本保存到文件中并打开此文件。

使用 AWS SCT 将 Teradata MultiLoad 作业脚本转换为 Amazon Redshift RSQL

您可以使用 AWS SCT 将 Teradata MultiLoad 作业脚本转换为 Amazon Redshift RSQL。

Teradata MultiLoad 作业脚本是一组用于批量维护 Teradata 数据库的命令。Teradata MultiLoad 导入任务可对最多五个不同的表和视图执行多种不同的插入、更新和删除操作。Teradata MultiLoad 删除任务可以从单个表中删除大量行。

AWS SCT 将 Teradata MultiLoad 命令和 SQL 语句转换为与 Amazon Redshift RSQL 兼容的格式。将 Teradata 数据库迁移到 Amazon Redshift 后，使用这些转换后的脚本管理 Amazon Redshift 数据库中的数据。

主题

将 MultiLoad 作业脚本添加到 AWS SCT 项目中

您可以将多个脚本添加到单个 AWS SCT 项目。

将 MultiLoad 作业脚本添加到 AWS SCT 项目中

1. 在 AWS SCT 中创建新项目，或打开现有项目。有关更多信息，请参阅the section called “创建项目”。
2. 从菜单中选择添加源，然后选择 Teradata 将源数据库添加到项目中。有关更多信息，请参阅将 Teradata 用作源。
3. 从菜单中选择添加目标，然后将目标 Amazon Redshift 数据库添加到 AWS SCT 项目中。

   您可以使用虚拟的 Amazon Redshift 目标数据库平台。有关更多信息，请参阅使用虚拟目标。
4. 创建新的映射规则，其中包括源 Teradata 数据库和 Amazon Redshift 目标。有关更多信息，请参阅添加新映射规则。
5. 在视图菜单上，选择主视图。
6. 在左侧面板中，展开脚本节点。
7. 选择 MultiLoad，打开上下文（右键单击）菜单，然后选择加载脚本。
8. 输入源 Teradata MultiLoad 作业脚本的位置，然后选择选择文件夹。

   AWS SCT 显示加载脚本窗口。
9. 请执行下列操作之一：

   - 如果 Teradata MultiLoad 作业脚本不包含替代变量，请选择无替代变量，然后选择确定将脚本添加到 AWS SCT 项目中。
   - 如果 Teradata MultiLoad 作业脚本包含替代变量，请配置替代变量。有关更多信息，请参阅在 MultiLoad 作业脚本中配置替代变量。

在 Teradata MultiLoad 作业脚本中使用 AWS SCT 配置替代变量

Teradata MultiLoad 作业脚本可能包含替代变量。例如，您可以使用带有替代变量的单个脚本将数据加载到不同的数据库。

运行带有替代变量的 MultiLoad 作业脚本之前，请确保为所有变量分配值。为此，您可以使用其他工具或应用程序，例如 Bash 脚本、UC4（Automic）等。

AWS SCT 只有在替代变量赋值后才能解析和转换替代变量。在开始转换源 Teradata MultiLoad 作业脚本之前，请确保为所有替代变量分配了值。您可以使用 AWS SCT 在 Teradata 脚本中配置替代变量。

在 MultiLoad 作业脚本中配置替代变量

1. 将源 Teradata MultiLoad 作业脚本添加到 AWS SCT 项目中时，请选择使用替代变量。有关添加这些脚本的更多信息，请参阅 将 MultiLoad 作业脚本添加到 AWS SCT 项目中。
2. 在定义变量格式中，输入与脚本中所有替代变量相匹配的正则表达式。

   例如，如果替代变量的名称以 ${ 开头且以 } 结尾，则使用 \$\{\w+\} 正则表达式。要匹配以美元符号或百分号开头的替代变量，请使用 \$\w+|\%\w+ 正则表达式。

   AWS SCT 中的正则表达式符合 Java 正则表达式语法。有关更多信息，请参阅 Java 文档中的 java.util.regex 类模式。
3. 选择确定，将脚本加载到 AWS SCT 项目中，然后选择确定关闭加载脚本窗口。
4. 选择变量可查看所有发现的替代变量及其值。
5. 在值中，输入替代变量的值。

使用 AWS SCT 转换 Teradata MultiLoad 作业脚本

接下来，了解如何使用 AWS SCT 将 Teradata MultiLoad 作业转换为 Amazon Redshift RSQL。

将 Teradata MultiLoad 作业脚本转换为 Amazon Redshift RSQL

1. 将 MultiLoad 作业脚本添加到 AWS SCT 项目中。有关更多信息，请参阅将 MultiLoad 作业脚本添加到 AWS SCT 项目中。
2. 配置替代变量并输入其值。有关更多信息，请参阅在 MultiLoad 作业脚本中配置替代变量。
3. 在左侧面板中，展开脚本节点。
4. 请执行下列操作之一：

   - 要转换单个 MultiLoad 作业脚本，请展开 MultiLoad 节点，选择要转换的脚本，然后从上下文（右键单击）菜单中选择转换脚本。
   - 要转换多个脚本，请确保选择所有要转换的脚本。选择 MultiLoad，打开上下文（右键单击）菜单，然后选择转换脚本。
5. 请执行下列操作之一：

   - 如果您将源数据文件存储在 Amazon S3 上，请选择源数据文件位置的 S3 对象路径。

为源数据文件输入 Amazon S3 存储桶文件夹和清单文件 Amazon S3 存储桶的值。

- 如果您未将源数据文件存储在 Amazon S3 上，请选择源数据文件位置中的主机地址。

  为源数据文件输入主机的 URL 或 IP 地址、主机用户登录名和清单文件的 Amazon S3 存储桶。

6. 选择确定。

   AWS SCT 将所有选定的 Teradata MultiLoad 作业脚本转换为与 Amazon Redshift RSQL 兼容的格式。在目标数据库面板的脚本节点中找到转换后的脚本。

7. 编辑转换后的 Amazon Redshift RSQL 脚本或将其保存。有关更多信息，请参阅编辑和保存转换后的 MultiLoad 作业脚本。

使用 AWS SCT 管理 Teradata MultiLoad 作业脚本

您可以添加多个 Teradata MultiLoad 作业脚本或从 AWS SCT 项目中移除一个 MultiLoad 作业脚本。

将新的 MultiLoad 作业脚本添加到 AWS SCT 项目中

1. 在左侧面板中，展开脚本节点。
2. 选择 MultiLoad 节点，然后打开上下文 (右键单击) 菜单。
3. 选择加载脚本。
4. 输入添加新的 MultiLoad 作业脚本和配置替代变量所需的信息。有关更多信息，请参阅 将 MultiLoad 作业脚本添加到 AWS SCT 项目中 和 在 MultiLoad 作业脚本中配置替代变量：

从 AWS SCT 项目中移除 MultiLoad 作业脚本

1. 展开左侧面板中脚本下的 MultiLoad 节点。
2. 选择要删除的脚本，然后打开上下文 (右键单击) 菜单。
3. 选择删除脚本。

使用 AWS SCT 创建 Teradata MultiLoad 作业脚本转换评估报告

MultiLoad 作业脚本转换评估报告提供了有关转换 MultiLoad 命令和 SQL 语句的信息。将源脚本转换为 Amazon Redshift RSQL 命令和 Amazon Redshift 的 SQL 语句。评估报告包括 AWS SCT 无法转换的 MultiLoad 命令和 SQL 语句的操作项。

创建 Teradata MultiLoad 作业脚本转换评估报告

1. 展开左侧面板中脚本下的 MultiLoad 节点。
2. 选择要创建评估报告的脚本，打开上下文（右键单击）菜单，然后选择创建报告。
3. 查看摘要选项卡。摘要选项卡显示 MultiLoad 作业脚本评估报告的执行摘要信息。它包括源脚本中所有 MultiLoad 命令和 SQL 语句的转换结果。
4. （可选）将 MultiLoad 作业脚本转换评估报告的本地副本另存为 PDF 文件或逗号分隔值 (CSV) 文件：

   - 要将 MultiLoad 作业脚本转换评估报告另存为 PDF 文件，请选择右上角的保存为 PDF。

     PDF 文件包含脚本转换执行摘要、操作项和建议。
   - 要将 MultiLoad 作业脚本转换评估报告另存为 CSV 文件，请选择右上角的保存为 CSV。

     AWS SCT 创建两个 CSV 文件。这些文件包含执行摘要、操作项目、推荐的操作以及转换脚本所需的估计人工操作的复杂性。
5. 选择操作项选项卡。此选项卡包含需要手动转换为 Amazon Redshift RSQL 的项目列表。从列表中选择一个操作项时，AWS SCT 会突出显示源 MultiLoad 作业脚本中该操作项适用的项。

使用 AWS SCT 编辑和保存转换后的 Teradata MultiLoad 作业脚本

您可以在 AWS SCT 项目的下部面板中编辑转换后的脚本。AWS SCT 将编辑后的脚本作为项目的一部分存储。

保存转换后的脚本

1. 展开目标数据库面板中脚本下的 RSQL 脚本节点。
2. 选择转换后的脚本，打开上下文（右键单击）菜单，然后选择保存脚本。
3. 输入用于保存转换后的脚本的文件夹路径，然后选择保存。

   AWS SCT 将转换后的脚本保存到文件中并打开此文件。

使用 AWS Schema Conversion Tool 迁移大数据框架

您可以使用 AWS Schema Conversion Tool（AWS SCT）将大数据框架迁移到 AWS Cloud。

目前，AWS SCT 支持将 Hadoop 集群迁移到 Amazon EMR 和 Amazon S3。此迁移过程包括 Hive 和 HDFS 服务。

此外，您还可以使用 AWS SCT 自动将 Apache Oozie 编排工作流程转换到 AWS Step Functions。

主题

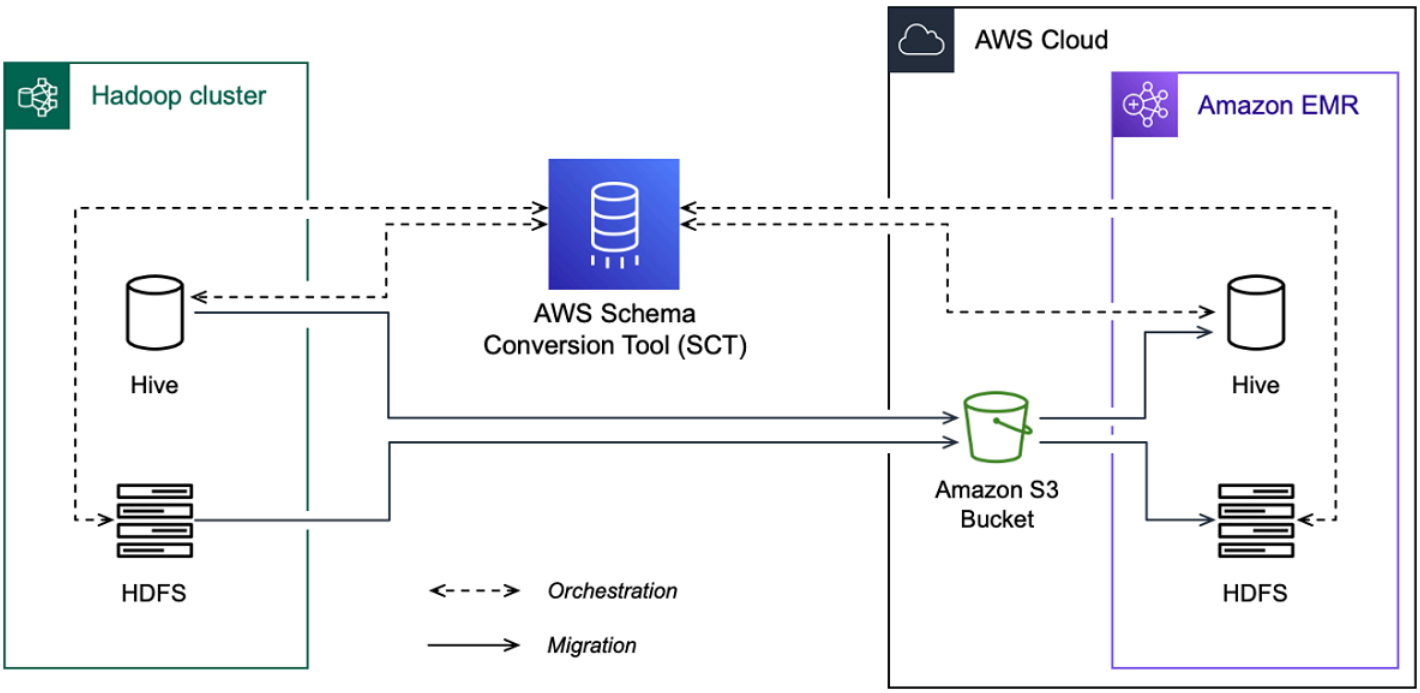
使用 AWS Schema Conversion Tool 将 Apache Hadoop 迁移到 Amazon EMR。

要迁移 Apache Hadoop 集群，请确保使用 AWS SCT 版本 1.0.670 或更高版本。另外，请熟悉 AWS SCT 的命令行界面（CLI）。有关更多信息，请参阅AWS SCT CLI 参考。

主题

迁移概述

下图显示了从 Apache Hadoop 迁移到 Amazon EMR 的架构图。



AWS SCT 将数据和元数据从源 Hadoop 集群迁移到 Amazon S3 存储桶。接下来，AWS SCT 使用源 Hive 元数据在目标 Amazon EMR Hive 服务中创建数据库对象。或者，您可以配置 Hive，从而将 AWS Glue Data Catalog 用作元存储。在这种情况下，AWS SCT 会将源 Hive 元数据迁移到 AWS Glue Data Catalog。

然后，您可以使用 AWS SCT 将数据从 Amazon S3 存储桶迁移到您的目标 Amazon EMR HDFS 服务。或者，您可以将数据保留在 Amazon S3 存储桶中，并将其用作 Hadoop 工作负载的数据存储库。

要启动 Hapood 迁移，您需要创建并运行 AWS SCT CLI 脚本。此脚本包含运行迁移的完整命令集。您可以下载和编辑 Hadoop 迁移脚本的模板。有关更多信息，请参阅获取 CLI 场景。

请确保您的脚本包含以下步骤，以便您可以运行从 Apache Hadoop 到 Amazon S3 和 Amazon EMR 的迁移。

步骤 1：连接到 Hadoop 集群

要开始迁移 Apache Hadoop 集群，请创建一个新 AWS SCT 项目。接下来，连接到源集群和目标集群。在开始迁移之前，请务必创建和配置目标 AWS 资源。

在此步骤中，您将使用以下 AWS SCT CLI 命令。

- `CreateProject`：创建新的项目
- `AddSourceCluster`：用于连接到 AWS SCT 项目中的源 Hadoop 集群。

- AddSourceClusterHive：连接到项目中的源 Hive 服务。
- AddSourceClusterHDFS：用于连接到项目中的源 HDFS 服务。
- AddTargetCluster：连接到项目中的目标 Amazon EMR 集群。
- AddTargetClusterS3：将 Amazon S3 存储桶添加到您的项目。
- AddTargetClusterHive：连接到项目中的目标 Hive 服务
- AddTargetClusterHDFS：连接到项目中的目标 HDFS 服务

有关使用这些 AWS SCT CLI 命令的示例，请参阅 使用 Apache Hadoop 作为源。

当您运行连接到源集群或目标集群的命令时，AWS SCT 会尝试与该集群建立连接。如果连接尝试失败，则 AWS SCT 停止运行 CLI 脚本中的命令并显示错误消息。

步骤 2：设置映射规则

连接到源集群和目标集群后，设置映射规则。映射规则定义了源集群的迁移目标。请务必为 AWS SCT 项目中添加的所有源集群设置映射规则。有关映射规则的更多信息，请参阅 在 AWS SCT 中创建映射规则。

在此步骤中，您将使用 AddServerMapping 命令。此命令使用两个参数，分别定义源集群和目标集群。您可以将该 AddServerMapping 命令与数据库对象的显式路径或对象名称一起使用。对于第一个选项，包括对象的类型及其名称。对于第二个选项，只包括对象名称。

- sourceTreePath：源数据库对象的显式路径。

  targetTreePath：目标数据库对象的显式路径。
- sourceNamePath：仅包含源对象名称的路径。

  targetNamePath：仅包含目标对象名称的路径。

以下代码示例使用源 testdb Hive 数据库和目标 EMR 集群的显式路径创建映射规则。

```
AddServerMapping
 -sourceTreePath: 'Clusters.HADOOP_SOURCE.HIVE_SOURCE.Databases.testdb'
 -targetTreePath: 'Clusters.HADOOP_TARGET.HIVE_TARGET'
/
```

你可以在 Windows 中使用此示例和以下示例。要在 Linux 中运行 CLI 命令，请确保根据您的操作系统相应地更新了文件路径。

以下代码示例使用仅包含对象名称的路径创建映射规则。

```
AddServerMapping
 -sourceNamePath: 'HADOOP_SOURCE.HIVE_SOURCE.testdb'
 -targetNamePath: 'HADOOP_TARGET.HIVE_TARGET'
/
```

您可以选择 Amazon EMR 或 Amazon S3 作为源对象的目标。对于每个源对象，在单个 AWS SCT 项目中您只能选择一个目标。要更改源对象的迁移目标，请删除现有的映射规则，然后创建新的映射规则。要删除映射规则，请使用 DeleteServerMapping 命令。此命令使用以下两个参数之一。

- sourceTreePath：源数据库对象的显式路径。
- sourceNamePath：仅包含源对象名称的路径。

有关 AddServerMapping 和 DeleteServerMapping 命令的更多信息，请参阅《AWS Schema Conversion Tool 参考》https://s3.amazonaws.com/publicsctdownload/AWS+SCT+CLI+Reference.pdf。

步骤 3：创建评估报告

在开始迁移之前，建议您创建一份评估报告。该报告总结了所有迁移任务，并详细说明了迁移期间将出现的操作项。为确保迁移不会失败，请在迁移之前查看此报告并解决操作项。有关更多信息，请参阅迁移评估报告。

在此步骤中，您将使用 CreateMigrationReport 命令。此命令使用两个参数。treePath 参数是必填的，而 forceMigrate 参数是可选的。

- treePath：保存评估报告副本的源数据库对象的明确路径。
- forceMigrate：如果设置为 true，即使项目包含引用同一对象的 HDFS 文件夹和 Hive 表，AWS SCT 也会继续迁移。默认值为 false。

您可以将评估报告的副本另存为 PDF 文件或逗号分隔值 (CSV) 文件。为此，使用 SaveReportPDF 或 SaveReportCSV 命令。

该 SaveReportPDF 命令将评估报告的副本另存为 PDF 文件。此命令使用四个参数。file 参数为必填的；其他参数是可选的。

- file：PDF 文件的路径及其名称。

- filter：您之前创建的筛选器的名称，用于定义要迁移的源对象的范围。
- treePath：保存评估报告副本的源数据库对象的明确路径。
- namePath：仅包含保存评估报告副本的目标对象名称的路径。

SaveReportCSV 命令将您的评估报告保存为三个 CSV 文件。此命令使用四个参数。directory 参数为必填的；其他参数是可选的。

- directory：AWS SCT 保存 CSV 文件的文件夹的路径。
- filter：您之前创建的筛选器的名称，用于定义要迁移的源对象的范围。
- treePath：保存评估报告副本的源数据库对象的明确路径。
- namePath：仅包含保存评估报告副本的目标对象名称的路径。

以下代码示例将评估报告的副本保存在 c:\sct\ar.pdf 文件中。

```
SaveReportPDF
 -file:'c:\sct\ar.pdf'
/
```

以下代码示例将评估报告的副本另存为 c:\sct 文件夹中的 CSV 文件。

```
SaveReportCSV
 -file:'c:\sct'
/
```

有关 SaveReportPDF 和 SaveReportCSV 命令的更多信息，请参阅《AWS Schema Conversion Tool CLI 参考》https://s3.amazonaws.com/publicsctdownload/AWS+SCT+CLI+Reference.pdf。

步骤 4：使用 AWS SCT 将 Apache Hadoop 集群迁移到 Amazon EMR

配置 AWS SCT 项目后，开始将本地 Apache Hadoop 集群迁移到 AWS Cloud。

在此步骤中，您将使用 Migrate、MigrationStatus 和 ResumeMigration 命令。

Migrate 命令会将源对象迁移到目标集群。此命令使用四个参数。请务必指定 filter 或 treePath 参数。其他参数都是可选的。

- filter：您之前创建的筛选器的名称，用于定义要迁移的源对象的范围。

- `treePath`：保存评估报告副本的源数据库对象的明确路径。
- `forceLoad`：如果设置为 `true`，AWS SCT 则会在迁移期间自动加载数据库元数据树。默认值为 `false`。
- `forceMigrate`：如果设置为 `true`，即使项目包含引用同一对象的 HDFS 文件夹和 Hive 表，AWS SCT 也会继续迁移。默认值为 `false`。

`MigrationStatus` 命令将返回有关命令进程的信息。要运行此命令，请为 `name` 参数输入迁移项目的名称。您在 `CreateProject` 命令中指定了此名称。

`ResumeMigration` 命令会恢复您使用 `Migrate` 命令启动的中断迁移。该 `ResumeMigration` 命令不使用参数。要恢复迁移，您必须连接到源集群和目标集群。有关更多信息，请参阅管理迁移项目。

以下代码示例将数据从您的源 HDFS 服务迁移到 Amazon EMR。

```
Migrate
 -treePath: 'Clusters.HADOOP_SOURCE.HDFS_SOURCE'
 -forceMigrate: 'true'
/
```

运行 CLI 脚本

编辑 AWS SCT CLI 脚本后，将其另存为 `.scts` 扩展名的文件。现在，您可以从 AWS SCT 安装路径的 `app` 文件夹中运行脚本。为此，请使用以下命令。

```
RunSCTBatch.cmd --pathtoscts "C:\script_path\hadoop.scts"
```

在前面的示例中，使用 CLI 脚本将 *`script_path`* 替换为您的文件路径。有关在 AWS SCT 中运行 CLI 脚本的更多信息，请参阅 脚本模式。

管理大数据迁移项目

完成迁移后，您可以保存和编辑 AWS SCT 项目以备将来使用。

要保存 AWS SCT 项目，请使用 `SaveProject` 命令。此命令不使用参数。

以下代码示例保存了您的 AWS SCT 项目。

```
SaveProject
```

```
/
```

要打开您的 AWS SCT 项目，请使用 OpenProject 命令。此命令使用一个必填参数。在 file 参数中，输入 AWS SCT 项目文件的路径及其名称。您在 CreateProject 命令中指定了项目名称。确保在项目文件名中添加 .scts 扩展名以运行 OpenProject 命令。

以下代码示例从 c:\sct 文件夹中打开 hadoop_emr 项目。

```
OpenProject
 -file: 'c:\sct\hadoop_emr.scts'
/
```

打开 AWS SCT 项目后，您无需添加源集群和目标集群，因为您已经将其添加到项目中。要开始使用源集群和目标集群，必须先连接到它们。为此，您可以使用 ConnectSourceCluster 和 ConnectTargetCluster 命令。这些命令使用的参数与 AddSourceCluster 和 AddTargetCluster 命令相同。您可以编辑 CLI 脚本并替换这些命令的名称，使参数列表保持不变。

以下代码示例连接到源 Hadoop 集群。

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

以下代码示例连接到目标 Amazon EMR 集群。

```
ConnectTargetCluster
 -name: 'HADOOP_TARGET'
 -vendor: 'AMAZON_EMR'
 -host: 'ec2-44-44-55-66.eu-west-1.EXAMPLE.amazonaws.com'
 -port: '22'
 -user: 'emr_user'
 -password: 'emr_password'
```

```
  -useSSL: 'true'
  -privateKeyPath: 'c:\path\name.pem'
  -passPhrase: '1234567890abcdef0!'
  -s3Name: 'S3_TARGET'
  -accessKey: 'AKIAIOSFODNN7EXAMPLE'
  -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
  -region: 'eu-west-1'
  -s3Path: 'doc-example-bucket/example-folder'
/
```

在前面的示例中，将 *hadoop_address* 替换为 Hadoop 集群的 IP 地址。如果需要，请配置端口变量的值。接下来，将 *hadoop_user* 和 *hadoop_password* 替换为用户的名字和该用户的密码。在*##\##*中，输入源 Hadoop 集群的 PEM 文件的名称和路径。有关添加源集群和目标集群的更多信息，请参阅 使用 Apache Hadoop 作为 AWS SCT 的源。

连接到源集群和目标 Hadoop 集群后，必须连接到 Hive 和 HDFS 服务以及 Amazon S3 存储桶。为此，您可以使用 `ConnectSourceClusterHive`、`ConnectSourceClusterHdfs`、`ConnectTargetClusterHive`、`Con` 和 `ConnectTargetClusterS3` 命令。这些命令使用的参数与您用于向项目添加 Hive 和 HDFS 服务以及 Amazon S3 存储桶的命令相同。编辑 CLI 脚本，将命令名称中的 `Add` 前缀替换为 `Connect`。

使用 AWS Schema Conversion Tool 将 Apache Oozie 转换为 AWS Step Functions

要转换 Apache Oozie 工作流程，请确保使用 AWS SCT 版本 1.0.671 或更高版本。另外，请熟悉 AWS SCT 的命令行接口 (CLI)。有关更多信息，请参阅AWS SCT CLI 参考。

主题

转换概述

Apache Oozie 源代码包括操作节点、控制流节点和作业属性。操作节点定义了 Apache Oozie 工作流程中运行的作业。当您使用 Apache Oozie 编排 Apache Hadoop 集群时，操作节点会包含一个 Hadoop 作业。控制流节点提供了一种控制工作流路径的机制。控制流节点包括 `start`、`end`、`decision`、`fork` 和 `join` 等节点。

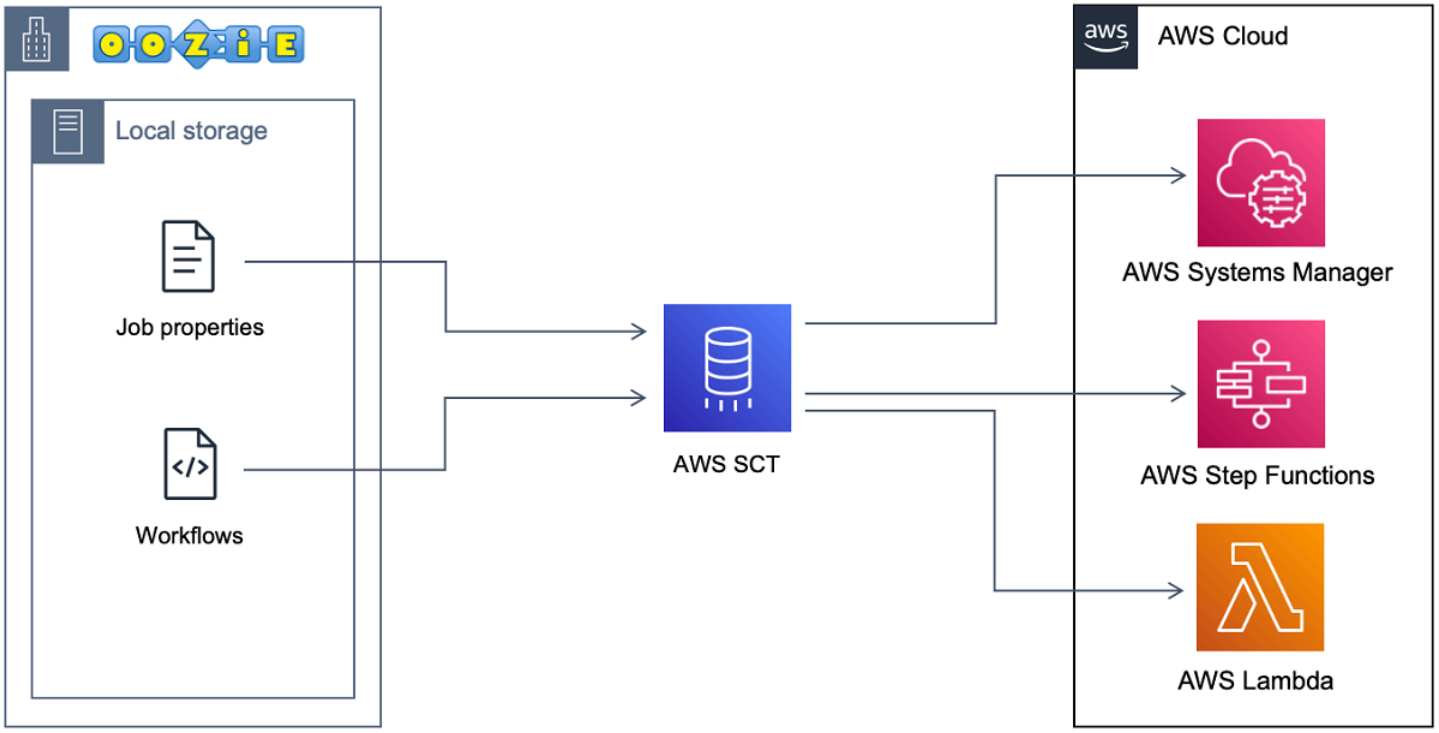
AWS SCT 将您的源操作节点和控制流节点转换为 AWS Step Functions。在 AWS Step Functions 中，您可以使用 Amazon States Language（ASL）定义工作流程。AWS SCT 使用 ASL 定义状态机和可执行工作的状态集合，确定要过渡到下一个状态的状态，并在出错时停止执行。接下来，AWS SCT 上传带有状态机定义的 JSON 文件。然后，AWS SCT 可以使用 AWS Identity and Access Management（IAM）角色在 AWS Step Functions 中配置状态机。有关更多信息，请参阅 AWS Step Functions 开发人员指南中的什么是 AWS Step Functions？

此外，AWS SCT 还会创建一个带有 AWS Lambda 函数的扩展包，次函数会模拟 AWS Step Functions 不支持的源函数。有关更多信息，请参阅使用 AWS SCT 扩展包。

AWS SCT 会将您的源作业属性迁移到 AWS Systems Manager。要存储参数名称和值，AWS SCT 使用 Parameter Store，AWS Systems Manager 的一项功能。有关更多信息，请参阅《AWS Systems Manager 用户指南》中的什么是 AWS Systems Manager？。

您可以使用 AWS SCT 自动更新参数的值和名称。由于 Apache Oozie 和 AWS Step Functions 之间的架构差异，您可能需要配置参数。AWS SCT 可以在源文件中找到指定的参数名称或值并将其替换为新值。有关更多信息，请参阅步骤 3：配置参数。

下图显示了 Apache Oozie 转换为 AWS Step Functions 的架构图。



要开始转换，请创建并运行 AWS SCT CLI 脚本。此脚本包含运行转换的完整命令集。你可以下载和编辑 Apache Oozie 转换脚本的模板。有关更多信息，请参阅获取 CLI 场景。

请确保脚本包含以下步骤。

步骤 1：连接到源服务和目标服务

要开始转换 Apache Oozie 集群，请创建一个新 AWS SCT 项目。接下来，连接到源服务和目标服务。在开始迁移之前，请务必创建和配置目标 AWS 资源。有关更多信息，请参阅使用 Apache Oozie 为源的先决条件。

在此步骤中，您将使用以下 AWS SCT CLI 命令。

- `CreateProject`：创建新的 AWS SCT 项目
- `AddSource`：在 AWS SCT 项目中添加源 Apache Oozie 文件。
- `ConnectSource`：作为源连接到 Apache Oozie。
- `AddTarget`：将 AWS Step Functions 作为迁移目标添加到项目中。
- `ConnectTarget`：连接到 AWS Step Functions。

有关使用这些 AWS SCT CLI 命令的示例，请参阅 使用 Apache Oozie 作为源。

当您运行 ConnectSource 或 ConnectTarget 命令时，AWS SCT 会尝试与您的服务建立连接。如果连接尝试失败，则 AWS SCT 停止运行 CLI 脚本中的命令并显示错误消息。

步骤 2：设置映射规则

连接到源服务和目标服务后，设置映射规则。映射规则定义源 Apache Oozie 工作流程和参数的迁移目标。有关映射规则的更多信息，请参阅 在 AWS SCT 中创建映射规则。

要定义要转换的源对象和目标对象，请使用 AddServerMapping 命令。此命令使用两个参数：sourceTreePath 和targetTreePath。这些参数值包括源对象和目标对象的显式路径。要使 Apache Oozie 进行 AWS Step Functions 转换，这些参数必须以 ETL 开头。

以下代码示例为 OOZIE 和 AWS_STEP_FUNCTIONS 对象创建映射规则。您在上一步中使用 AddSource 和 AddTarget 命令将这些对象添加到 AWS SCT 项目中。

```
AddServerMapping
    -sourceTreePath: 'ETL.APACHE_OOZIE'
    -targetTreePath: 'ETL.AWS_STEP_FUNCTIONS'
/
```

有关 AddServerMapping 命令的更多信息，请参阅《AWS Schema Conversion Tool CLI 参考》https://s3.amazonaws.com/publicsctdownload/AWS+SCT+CLI+Reference.pdf。

步骤 3：配置参数

如果您的源 Apache Oozie 工作流程使用参数，则可能需要在转换为 AWS Step Functions 后更改其值此外，您可能需要添加新参数以与 AWS Step Functions 一起使用。

在此步骤中，您将使用 AddParameterMapping 和 AddTargetParameter 命令。

要替换源文件中的参数值，请使用 AddParameterMapping 命令。AWS SCT 会扫描源文件，按名称或值查找参数，然后更改其值。您可以运行单个命令扫描所有源文件。您可以使用以下列表中的前三个参数之一定义要扫描的文件范围。此命令最多使用六个参数。

- filterName：源对象的过滤器名称。您可以使用 CreateFilter 命令创建过滤器。
- treePath：源对象的显式路径。
- namePath：特定源对象的显式路径。
- sourceParameterName：源参数的名称。

- sourceValue：源参数的值。
- targetValue：目标参数的值。

以下代码示例将值等于 c:\oozie\hive.py 的所有参数替换为 s3://bucket-oozie/hive.py 值。

```
AddParameterMapping
 -treePath: 'ETL.OOZIE.Applications'
 -sourceValue: 'c:\oozie\hive.py'
 -targetValue: 's3://bucket-oozie/hive.py'
/
```

以下代码示例将名称等于 nameNode 的所有参数替换为 hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020 值。

```
AddParameterMapping
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -targetValue: 'hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020'
/
```

以下代码示例将名称等于 nameNode 且值等于 hdfs://ip-55.eu-west-1.compute.internal:8020 的所有参数替换为 targetValue 参数中的值。

```
AddParameterMapping
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -sourceValue: 'hdfs://ip-55-66-77-88.eu-west-1.compute.internal:8020'
    -targetValue: 'hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020'
/
```

除了源文件中的现有参数外，要在目标文件中添加新参数，请使用 AddTargetParameter 命令。此命令使用与 AddParameterMapping 命令相同的参数集。

以下代码示例添加了 clusterId 目标参数而不是 nameNode 参数。

```
AddTargetParameter
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -sourceValue: 'hdfs://ip-55-66-77-88.eu-west-1.compute.internal:8020'
```

```
    -targetParameter: 'clusterId'
    -targetValue: '1234567890abcdef0'
/
```

有关 AddServerMapping、AddParameterMapping、AddTargetParameter 和 CreateFilter 的更多信息，请参阅《AWS Schema Conversion Tool CLI 参考》https://s3.amazonaws.com/publicsctdownload/AWS+SCT+CLI+Reference.pdf。

步骤 4：创建评估报告

在开始转换之前，建议您创建一份评估报告。该报告总结了所有迁移任务，并详细说明了迁移期间将出现的操作项。为确保迁移不会失败，请在迁移之前查看此报告并解决操作项。有关更多信息，请参阅迁移评估报告。

在此步骤中，您将使用 CreateReport 命令。此命令使用两个参数。第一个参数描述了 AWS SCT 为其创建评估报告的源对象。为此，请使用以下参数之一：filterName、treePath 或 namePath。此参数是必填的。此外，您还可以添加可选的布尔参数 forceLoad。如果将此参数设置为 true，则 AWS SCT 会自动加载 CreateReport 命令中指定的源对象的所有子对象。

以下代码示例为源 Oozie 文件的 Applications 节点创建评估报告。

```
CreateReport
    -treePath: 'ETL.APACHE_OOZIE.Applications'
/
```

您可以将评估报告的副本另存为 PDF 文件或逗号分隔值 (CSV) 文件。为此，使用SaveReportPDF 或 SaveReportCSV 命令。

该 SaveReportPDF 命令将评估报告的副本另存为 PDF 文件。此命令使用四个参数。file 参数为必填的；其他参数是可选的。

- file：PDF 文件的路径及其名称。
- filter：您之前创建的筛选器的名称，用于定义要迁移的源对象的范围。
- treePath：保存评估报告副本的源数据库对象的明确路径。
- namePath：仅包含保存评估报告副本的目标对象名称的路径。

SaveReportCSV 命令将您的评估报告保存为 CSV 文件。此命令使用四个参数。directory 参数为必填的；其他参数是可选的。

- directory：AWS SCT 保存 CSV 文件的文件夹的路径。
- filter：您之前创建的筛选器的名称，用于定义要迁移的源对象的范围。
- treePath：保存评估报告副本的源数据库对象的明确路径。
- namePath：仅包含保存评估报告副本的目标对象名称的路径。

以下代码示例将评估报告的副本保存在 c:\sct\ar.pdf 文件中。

```
SaveReportPDF
 -file:'c:\sct\ar.pdf'
/
```

以下代码示例将评估报告的副本另存为 c:\sct 文件夹中的 CSV 文件。

```
SaveReportCSV
 -file:'c:\sct'
/
```

有关CreateReport、SaveReportPDF 和 SaveReportCSV 命令的更多信息，请参阅《AWS Schema Conversion Tool CLI 参考》https://s3.amazonaws.com/publicsctdownload/AWS+SCT+CLI+Reference.pdf。

第 5 步：用 AWS SCT 将 Apache Oozie 工作流程转换为 AWS Step Functions

配置 AWS SCT 项目后，转换源代码并将其应用于 AWS Cloud。

在此步骤中，您将使用 Convert、SaveOnS3、ConfigureStateMachine 和 ApplyToTarget 命令。

Migrate 命令会将源对象迁移到目标集群。此命令使用四个参数。请务必指定 filter 或 treePath 参数。其他参数都是可选的。

- filter：您之前创建的筛选器的名称，用于定义要迁移的源对象的范围。
- namePath：特定源对象的显式路径。
- treePath：保存评估报告副本的源数据库对象的明确路径。
- forceLoad：如果设置为 true，AWS SCT 则会在迁移期间自动加载数据库元数据树。默认值为 false。

以下代码示例转换源 Oozie 文件中 Applications 文件夹中的文件。

```
Convert
    -treePath: 'ETL.APACHE_OOZIE.Applications'
/
```

SaveOnS3 会将状态机定义上传到 Amazon S3 存储桶。此命令使用 treePath 参数。要运行此命令，请使用带有状态机定义的目标文件夹作为此参数的值。

以下内容将 AWS_STEP_FUNCTIONS 目标对象的 State machine definitions 文件夹上传到 Amazon S3 存储桶。AWS SCT 将使用您在 先决条件 步骤中的 AWS 服务配置文件中存储的 Amazon S3 存储桶。

```
SaveOnS3
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machine definitions'
/
```

ConfigureStateMachine 命令配置状态机。此命令最多使用六个参数。请务必使用以下列表中前三个参数中的一个来定义目标范围。

- filterName：目标对象的过滤器名称。您可以使用 CreateFilter 命令创建过滤器。
- treePath：目标对象的显式路径。
- namePath：特定目标对象的显式路径。
- iamRole：提供对步骤机访问的 IAM 角色的 Amazon 资源名称（ARN）。此参数为必需参数。

以下代码示例使用 *role_name* IAM 角色配置 AWS_STEP_FUNCTIONS 中定义的状态机。

```
ConfigureStateMachine
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machine definitions'
    -role: 'arn:aws:iam::555555555555:role/role_name'
/
```

ApplyToTarget 命令将转换后的代码应用于目标服务器。要运行此命令，请使用以下参数之一：filterName、treePath 或 namePath 来定义要应用的目标对象。

以下代码示例将 app_wp 状态机应用于 AWS Step Functions。

```
ApplyToTarget
```

```
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machines.app_wp'
/
```

要确保转换后的代码产生与源代码相同的结果，您可以使用 AWS SCT 扩展包。这是一组模拟 AWS Step Functions 不支持的 Apache Oozie 函数的函数。要安装此扩展包，可以使用 `CreateLambdaExtPack` 命令。

此命令最多使用五个参数。确保将 **`Oozie2SF`** 用于 `extPackId`。在本例中，AWS SCT 为源 Apache Oozie 函数创建一个扩展包。

- `extPackId`：一组 Lambda 函数的唯一标识符。此参数为必需参数。
- `tempDirectory`：AWS SCT 可以存储临时文件的路径。此参数为必需参数。
- `awsProfile`：AWS 配置文件的名称。
- `lambdaExecRoles`：用于 Lambda 函数的执行角色的 Amazon 资源名称（ARN）列表。
- `createInvokeRoleFlag`：表示是否为其创建 AWS Step Functions 执行角色的布尔标志。

要安装和使用扩展包，请确保您提供所需的权限。有关更多信息，请参阅使用扩展包中 AWS Lambda 函数的权限。

有关 `Convert`、`SaveOnS3`、`ConfigureStateMachine`、`ApplyToTarget` 和 `CreateLambdaExtPack` 的更多信息，请参阅《AWS Schema Conversion Tool CLI 参考》https://s3.amazonaws.com/publicsctdownload/AWS+SCT+CLI+Reference.pdf。

运行 CLI 脚本

编辑 AWS SCT CLI 脚本后，将其另存为 `.scts` 扩展名的文件。现在，您可以从 AWS SCT 安装路径的 `app` 文件夹中运行脚本。为此，请使用以下命令。

```
RunSCTBatch.cmd --pathtoscts "C:\script_path\oozie.scts"
```

在前面的示例中，使用 CLI 脚本将 *`script_path`* 替换为您的文件路径。有关在 AWS SCT 中运行 CLI 脚本的更多信息，请参阅 脚本模式。

AWS SCT 可以转换为 AWS Step Functions 的 Apache Oozie 节点

您可以使用 AWS SCT 将 Apache Oozie 操作节点和控制流节点转换为 AWS Step Functions。

支持的操作节点包括：

- Hive 操作
- Hive2 操作
- Spark 操作
- MapReduce 流操作
- Java 操作
- DistCp 操作
- Pig 操作
- Sqoop 操作
- FS 操作
- Shell 操作

支持的控制流节点包括：

- 开始操作
- 结束操作
- 终止操作
- 决策操作
- 分流操作
- 加入操作

配合使用 AWS SCT 和 AWS DMS

将 AWS SCT 复制代理与 AWS DMS 结合使用

对于非常大的数据库迁移，您可以使用 AWS SCT 复制代理（aws-schema-conversion-tool-dms-agent）将数据从本地数据库复制到 Amazon S3 或 AWS Snowball Edge 设备。复制代理可与 AWS DMS 结合使用，并且在 AWS SCT 关闭的情况下，它可以在后台工作。

当与 AWS Snowball Edge 结合使用时，AWS SCT 代理会将数据复制到 AWS Snowball 设备。然后将该设备发送到 AWS，并将数据加载到 Amazon S3 存储桶。在这段时间内，AWS SCT 代理将继续运行。然后，该代理会在 Amazon S3 上接收数据，并将数据复制到目标终端节点。

有关更多信息，请参阅将本地数据仓库中的数据迁移到 Amazon Redshift。

将 AWS SCT 数据提取代理与 AWS DMS 结合使用

在 AWS SCT 中，您可以找到一个数据提取代理（`aws-schema-conversion-tool-extractor`），它可以帮助您更轻松地从 Apache Cassandra 迁移到 Amazon DynamoDB。Cassandra 和 DynamoDB 都是 NoSQL 数据库，但它们的系统架构和数据表示形式有所不同。您可以在 AWS SCT 中使用基于向导的工作流程自动执行 Cassandra 到 DynamoDB 的迁移过程。AWS SCT 与 AWS Database Migration Service（AWS DMS）集成以执行实际迁移。

有关更多信息，请参阅将本地数据仓库中的数据迁移到 Amazon Redshift。

配合使用 AWS SCT 和 AWS DMS 时提高日志记录级别

配合使用 AWS SCT 和 AWS DMS 时，您可以提高日志记录级别，例如，您需要使用 AWS Support。

AWS SCT 和所需的驱动程序安装完成后，通过选择 AWS SCT 图标打开应用程序。如果您看到更新通知，则可以选择在项目完成之前或之后进行更新。如果自动项目窗口打开，请关闭该窗口并手动创建项目。

配合使用 AWS SCT 和 AWS DMS 时提高日志记录级别

1. 在设置菜单，选择全局设置。
2. 在全局设置窗口中，选择日志记录。
3. 对于调试模式，请选择 True。

4. 在消息级别部分，您可以修改以下类型的日志：

   - 常规
   - 加载程序
   - 解析器
   - 打印机
   - 解析程序
   - 遥测
   - 转换器

   默认情况下，所有消息级别都设置为 Info。
5. 为要更改的任何消息级别类型选择日志记录级别：

   - 跟踪（最详细的日志记录）
   - Debug
   - Info
   - 警告
   - 错误（最不详细的日志记录）
   - 重大
   - 必需
6. 选择应用即可修改项目的设置。
7. 选择确定即可关闭全局设置对话框。

将本地数据仓库中的数据迁移到 Amazon Redshift

您可以使用 AWS SCT 代理从本地数据仓库中提取数据并将其迁移到 Amazon Redshift。代理会提取您的数据并将数据上传到 Amazon S3，或者如果要进行大规模迁移，则上传到 AWS Snowball Edge 设备。然后，您可以使用 AWS SCT 代理将数据复制到 Amazon Redshift。

或者，您可以使用 AWS Database Migration Service (AWS DMS) 将数据迁移到亚马逊 Redshift。AWS DMS 的优点是支持持续复制（更改数据捕获）。但是，要提高数据迁移的速度，可以并行使用多个 AWS SCT 代理。根据我们的测试， AWS SCT 代理迁移数据的速度超过 AWS DMS 15-35%。速度的差异是由于数据压缩、支持并行迁移表分区以及不同的配置设置造成的。有关更多信息，请参阅将 Amazon Redshift 数据库用作 AWS Database Migration Service的目标。

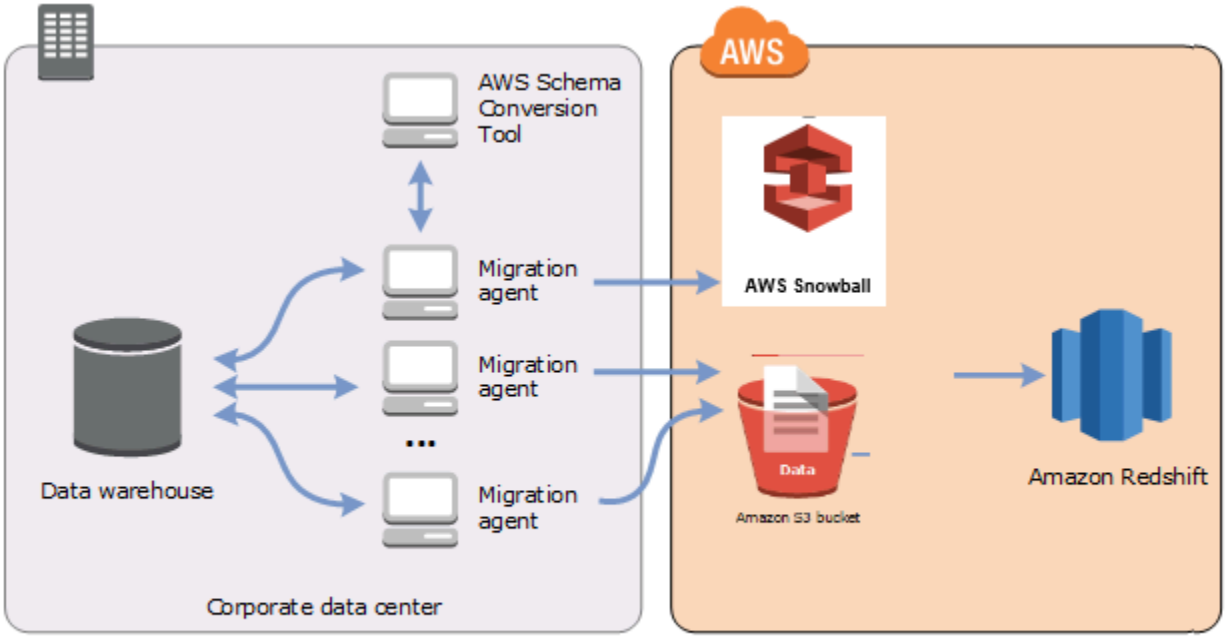
Amazon S3 是一种存储和检索服务。要将数据元存储到 Amazon S3 中，请将要存储的文件上传到 Amazon S3 存储桶中。在上传文件时，您可以设置对象以及任何元数据的权限。

大规模迁移

大规模的数据迁移可能包括许多 TB 的信息，并且可能会因为网络性能和必须移动的数据量庞大而减慢。 AWS Snowball Edge 是一项 AWS 服务，您可以使用 AWS自有设备将数据 faster-than-network 快速传输到云端。一台 AWS Snowball Edge 设备最多可以容纳 100 TB 的数据。它使用 256 位加密和行业标准的可信平台模块 (TPM)，以确保您的数据既安全又完整 chain-of-custody 。 AWS SCT 可与 AWS Snowball Edge 设备配合使用。

当您使用 AWS SCT 和 AWS Snowball Edge 设备时，您可以分两个阶段迁移数据。首先，您可以使用 AWS SCT 在本地处理数据，然后将该数据移至 AWS Snowball Edge 设备。然后，您可以 AWS 使用 AWS Snowball Edge 流程将设备发送到，然后 AWS 自动将数据加载到 Amazon S3 存储桶中。接下来，当数据在 Amazon S3 上可用时，您可以使用将数据迁移 AWS SCT 到 Amazon Redshift。关闭时 AWS SCT ，数据提取代理可以在后台运行。

下图显示了支持的方案。



以下源数据仓库当前支持数据提取代理：

- Azure Synapse Analytics
- BigQuery
- Greenplum 数据库（版本 4.3）
- Microsoft SQL Server（版本 2008 及更高版本）
- Netezza（版本 7.0.3 及更高版本）
- Oracle（版本 10 及更高版本）
- Snowflake（版本 3）
- Teradata（版本 13 及更高版本）
- Vertica（版本 7.2.2 及更高版本）

如果您需要符合联邦信息处理标准（FIPS）安全要求，则可以连接到 Amazon Redshift 的 FIPS 端点。FIPS 终端节点可在以下 AWS 区域使用：

- 美国东部（弗吉尼亚州北部）区域（redshift-fips.us-east-1.amazonaws.com）
- 美国东部（俄亥俄州）区域（redshift-fips.us-east-2.amazonaws.com）
- 美国西部（北加利福尼亚）区域（redshift-fips.us-west-1.amazonaws.com）

- 美国西部（俄勒冈州）区域（redshift-fips.us-west-2.amazonaws.com）

使用以下主题中的信息了解如何使用数据提取代理。

主题

使用数据提取代理的先决条件

在使用数据提取代理之前，请为 Amazon Redshift 用户添加作为目标的 Amazon Redshift 所需的权限。有关更多信息，请参阅 将 Amazon Redshift 作为目标的权限。

然后，存储 Amazon S3 存储桶信息并设置安全套接字层 (SSL) 信任和密钥存储。

Amazon S3 设置

当代理提取数据后，它们会将其上传到 Amazon S3 存储桶。在继续操作之前，您必须提供凭证才能连接到您的 AWS 账户和 Amazon S3 存储桶。您将凭证和存储桶信息存储在全局应用程序设置的配置文

件中，然后将该配置文件与您的 AWS SCT 项目关联。如有必要，请选择全局设置即可创建新的配置文件。有关更多信息，请参阅 将 AWS 服务配置文件存储在 AWS SCT 中。

要将数据迁移到您的目标 Amazon Redshift 数据库， AWS SCT 数据提取代理需要获得代表您访问 Amazon S3 存储桶的权限。要提供此权限，请使用以下策略创建一个 AWS Identity and Access Management (IAM) 用户。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:PutObjectTagging"
            ],
            "Resource": [
                "arn:aws:s3:::bucket_name/*",
                "arn:aws:s3:::bucket_name"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::bucket_name"
            ],
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Action": [
                "iam:GetUser"
            ],
```

```
            "Resource": [
                "arn:aws:iam::111122223333:user/DataExtractionAgentName"
            ],
            "Effect": "Allow"
        }
    ]
}
```

在上面示例中，请将 *bucket_name* 替换为 Amazon S3 存储桶名称。然后，将 *111122223333:user/DataExtractionAgentName* 替换为 IAM 用户的名称。

假定 IAM 角色

为了提高安全性，您可以使用 AWS Identity and Access Management (IAM) 角色访问您的 Amazon S3 存储桶。为此，请在没有任何权限的情况下为数据提取代理创建一个 IAM 用户。然后，创建一个启用 Amazon S3 访问权限的 IAM 角色，并指定服务和可以担任此角色的用户列表。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

配置 IAM 角色以访问 Amazon S3 存储桶

1. 创建新的 IAM 用户 对于用户凭证，请选择编程访问类型。
2. 配置主机环境，以便您的数据提取代理可以扮演 AWS SCT 提供的角色。确保您在上一步中配置的用户允许数据提取代理使用凭证提供程序链。有关更多信息，请参阅《AWS SDK for Java 开发人员指南》中的使用凭证。
3. 创建一个有 Amazon S3 存储桶访问权限的新 IAM 角色。
4. 修改此角色的“信任”部分，以信任您之前创建的担任该角色的用户。在以下示例中，将 *111122223333:user/DataExtractionAgentName* 替换为用户名称。

   ```
   {
       "Effect": "Allow",
       "Principal": {
           "AWS": "arn:aws:iam::111122223333:user/DataExtractionAgentName"
       },
       "Action": "sts:AssumeRole"
   }
   ```

5. 将此角色的信任部分修改为信任 `redshift.amazonaws.com` 以担任该角色。

   ```
   {
       "Effect": "Allow",
   ```

```
    "Principal": {
        "Service": [
            "redshift.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole"
}
```

6. 将此角色附加到 Amazon Redshift 集群。

现在，您可以在 AWS SCT中运行数据提取代理。

当您使用 IAM 角色代入时，数据迁移的工作方式如下。数据提取代理启动并使用凭证提供程序链获取用户凭证。接下来，在中创建数据迁移任务 AWS SCT，然后指定数据提取代理要担任的 IAM 角色，然后启动任务。 AWS Security Token Service (AWS STS) 生成用于访问 Amazon S3 的临时证书。数据提取代理使用这些凭证将数据上传到 Amazon S3。

然后，为 Amazon Redshift AWS SCT 提供 IAM 角色。反过来，亚马逊 Redshift 会从中获得访问亚马逊 S AWS STS 3 的新临时证书。Amazon Redshift 使用这些凭证将数据从 Amazon S3 复制到 Amazon Redshift 表。

安全设置

AWS Schema Conversion Tool 和提取代理可以通过安全套接字层 (SSL) 进行通信。要启用 SSL，请设置信任存储和密钥存储。

建立与提取代理的安全通信

1. 启动 AWS Schema Conversion Tool.
2. 打开设置菜单，然后选择全局设置。此时显示 Global settings 对话框。
3. 选择安全性。
4. 选择生成信任存储和密钥存储，或选择选择现有信任存储。

   如果您选择生成信任存储和密钥存储，则可为信任存储和密钥存储指定名称和密码，并为生成的文件指定位置路径。您会在后面的步骤中用到这些文件。

   如果选择选择现有信任存储，则为信任存储和密钥存储指定密码和文件名。您会在后面的步骤中用到这些文件。
5. 指定信任存储和密钥存储后，请选择确定以关闭全局设置对话框。

配置数据提取代理环境

可以在一台主机上安装多个数据提取代理。但是，建议您在一台主机上运行一个数据提取代理。

要运行数据提取代理，请确保使用的主机至少有四个 vCPU 和 32 GB 内存。此外，将可用的最小内存设置为 AWS SCT 至少 4 GB。有关更多信息，请参阅 配置额外的内存。

代理主机的最佳配置和数量取决于每个客户的具体情况。请务必考虑诸如要迁移的数据量、网络带宽、提取数据的时间等因素。您可以先执行概念验证（PoC），然后根据此 PoC 的结果配置数据提取代理和主机。

安装提取代理

我们建议您将多个提取代理安装在不同的计算机上，与运行 AWS Schema Conversion Tool的计算机分开。

以下操作系统当前支持提取代理：

- Microsoft Windows
- Red Hat Enterprise Linux (RHEL) 6.0
- Ubuntu Linux（版本 14.04 及更高版本）

使用以下过程安装提取代理。对每台要安装提取代理的计算机重复此过程。

安装提取代理

1. 如果您尚未下载 AWS SCT 安装程序文件，请按照中的说明安装、验证和更新 AWS SCT进行下载。包含 AWS SCT 安装程序文件的.zip 文件还包含解压缩代理安装程序文件。
2. 下载并安装最新版本的 Amazon Corretto 11。有关更多信息，请参阅《Amazon Corretto 11 用户指南》中适用于 Amazon Corretto 11 的下载内容。。
3. 在一个名为 agents 的子文件夹中查找提取代理的安装程序文件。对于每个计算机操作系统，用于安装提取代理的正确文件如下所示。

操作系统	文件名
Microsoft Windows	`aws-schema-conversion-tool-extractor-2.0.1.` *`build-number`* `.msi`

操作系统	文件名
RHEL	aws-schema-conversion-tool-extractor -2.0.1. *build-number* .x86_64.rpm
Ubuntu Linux	aws-schema-conversion-tool-extractor -2.0.1. *build-number* .deb

4. 通过将安装程序文件复制到新计算机上，在另一台计算机上安装提取代理。
5. 运行安装程序文件。使用适合您的操作系统的说明，如下所示。

操作系统	安装说明
Microsoft Windows	双击该文件以运行安装程序。

RHEL	在下载或存储该文件的文件夹中运行以下命令：
	`sudo rpm -ivh aws-schema-conversion-tool-extractor -2.0.1. build-number .x86_64.rpm`
	`sudo ./sct-extractor-setup.sh --config`

Ubuntu Linux	在下载或存储该文件的文件夹中运行以下命令：
	`sudo dpkg -i aws-schema-conversion-tool-extractor -2.0.1. build-number .deb`
	`sudo ./sct-extractor-setup.sh --config`

6. 选择下一步，接受许可协议，然后选择下一步。
7. 输入 AWS SCT 数据提取代理的安装路径，然后选择下一步。
8. 选择安装以安装您的数据提取代理。

   AWS SCT 安装您的数据提取代理。要完成安装，请配置您的数据提取代理。 AWS SCT 自动启动配置设置程序。有关更多信息，请参阅 配置提取代理。
9. 配置数据提取代理后，选择完成以关闭安装向导。

配置提取代理

使用以下过程配置提取代理。对每台已安装提取代理的计算机重复此过程。

配置提取代理

1. 启动配置设置程序：

   - 在 Windows 中，在安装数据提取代理的过程中自动 AWS SCT 启动配置设置程序。

     根据需要，您可以手动启动设置程序。为此，请在 Windows 中运行 `ConfigAgent.bat` 文件。可以在安装代理的文件夹中找到此文件。
   - 在 RHEL 和 Ubuntu 中，从安装代理的位置运行 `sct-extractor-setup.sh` 文件。

   设置程序会提示您输入信息。对于每个提示，系统会显示默认值。
2. 在每次提示时接受默认值，或输入新值。

   指定以下信息：

   - 对于侦听端口，请输入代理将要侦听的端口号。
   - 对于添加源供应商，输入是，然后输入您的源数据仓库平台。
   - 对于 JDBC 驱动程序，输入安装了 JDBC 驱动程序的位置。
   - 在“工作文件夹”中，输入 AWS SCT 数据提取代理存储提取数据的路径。工作文件夹可能与代理分别位于不同的计算机，且单个工作文件夹可由不同计算机上的多个代理共享。
   - 对于启用 SSL 通信，请输入是。
   - 对于密钥存储，请输入密钥存储文件的位置。
   - 对于秘钥存储密码，请输入密钥存储的密码。
   - 对于启用客户端 SSL 身份验证，请输入是。
   - 对于信任存储，请输入信任存储文件的位置。
   - 对于信任存储密码，请输入信任存储的密码。

设置程序将更新提取代理的设置文件。设置文件名为 `settings.properties`，位于安装了提取代理的位置。

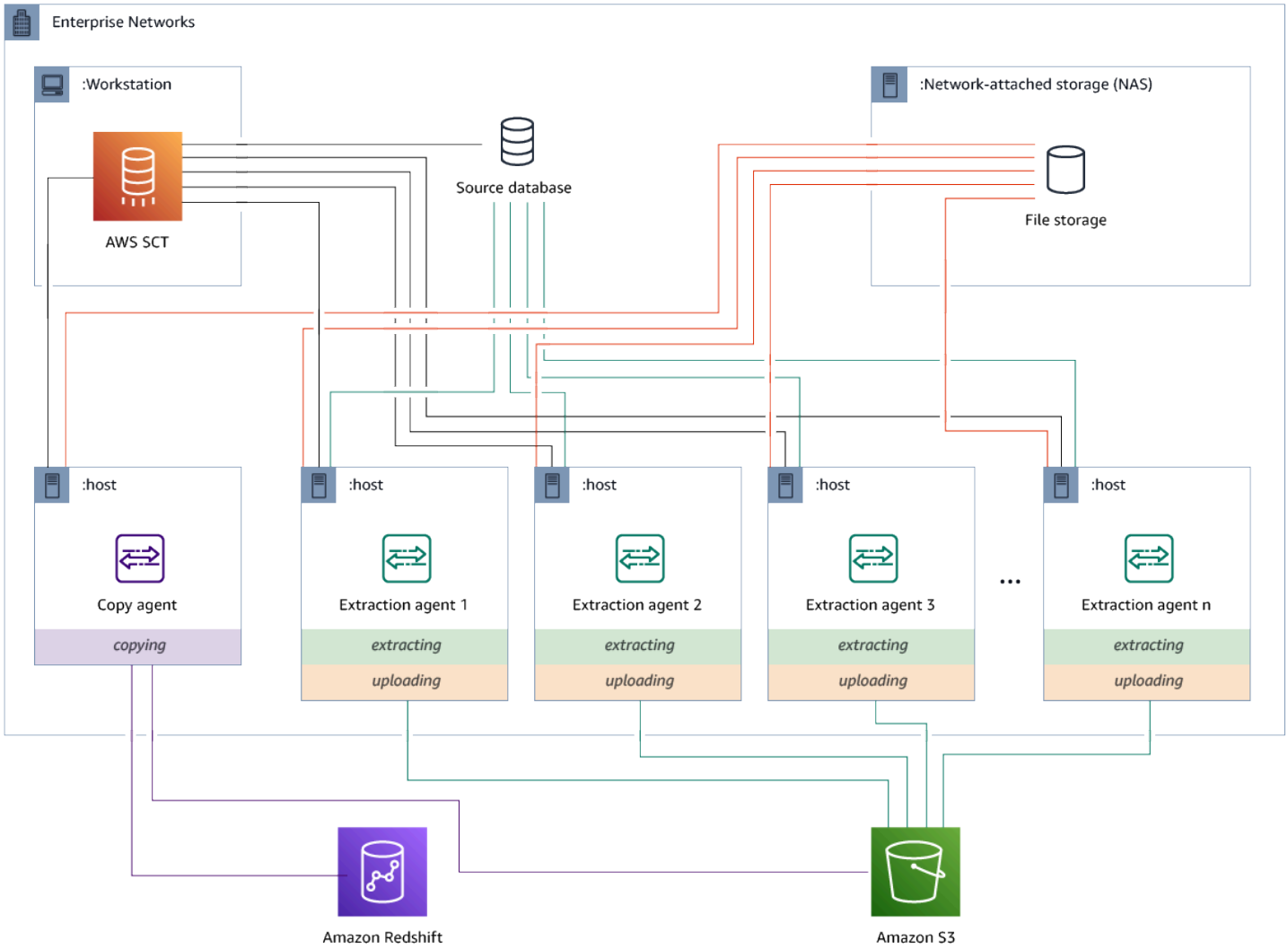
下面是一个设置文件示例。

```
$ cat settings.properties
#extractor.start.fetch.size=20000
#extractor.out.file.size=10485760
#extractor.source.connection.pool.size=20
#extractor.source.connection.pool.min.evictable.idle.time.millis=30000
#extractor.extracting.thread.pool.size=10
vendor=TERADATA
driver.jars=/usr/share/lib/jdbc/terajdbc4.jar
port=8192
redshift.driver.jars=/usr/share/lib/jdbc/RedshiftJDBC42-1.2.43.1067.jar
working.folder=/data/sct
extractor.private.folder=/home/ubuntu
ssl.option=OFF
```

要更改配置设置，可以使用文本编辑器编辑 settings.properties 文件或再次运行代理配置。

使用专用的复制代理安装和配置提取代理

可以在具有共享存储空间和专用复制代理的配置中安装提取代理。下图阐明了此方案。



当源数据库服务器支持多达 120 个连接且您的网络连接了充足的存储空间时，该配置可能很有用。使用以下过程配置具有专用复制代理的提取代理。

安装和配置提取代理和专用复制代理

1. 确保所有提取代理的工作目录使用共享存储空间上的相同文件夹。
2. 按照 安装提取代理 中的步骤安装提取代理。
3. 按照 配置提取代理 中的步骤配置提取代理，但仅指定源 JDBC 驱动程序。
4. 按照 配置提取代理 中的步骤配置专用复制代理，但仅指定 Amazon Redshift JDBC 驱动程序。

启动提取代理

使用以下过程启动提取代理。对每台已安装提取代理的计算机重复此过程。

提取代理充当侦听器。通过此过程启动代理时，代理将启动侦听以获得指示。在后面的部分中，您将发送代理指示，以从数据仓库提取数据。

启动提取代理

- 在安装了提取代理的计算机上，运行下列适用于您的操作系统的命令。

操作系统	启动命令
Microsoft Windows	双击 `StartAgent.bat` 批处理文件。
RHEL	在安装了代理的文件夹的路径中运行以下命令： `sudo initctl start sct-extractor`
Ubuntu Linux	在安装了代理的文件夹的路径中运行以下命令。使用适合您的 Ubuntu 版本的命令。 Ubuntu 14.04: `sudo initctl start sct-extractor` Ubuntu 15.04 及更高版本：`sudo systemctl start sct-extractor`

要检查代理的状态，请运行同一命令，但将 `start` 替换为 `status`。

要停止代理，请运行同一命令，但将 `start` 替换为 `stop`。

在中注册提取剂 AWS Schema Conversion Tool

您可以通过使用来管理提取剂 AWS SCT。提取代理充当侦听器。当他们收到来自的指令时 AWS SCT，他们会从您的数据仓库中提取数据。

使用以下步骤在您的 AWS SCT 项目中注册提取剂。

注册提取代理

1. 启动 AWS Schema Conversion Tool，然后打开一个项目。
2. 打开视图菜单，然后选择数据迁移视图（其他）。此时显示 Agents 选项卡。如果您之前注册过代理，则 AWS SCT 会在选项卡顶部的网格中显示这些代理。
3. 选择 Register。

   在项目中注册代理后，您无法在其他 AWS SCT 项目中注册同一个代理。如果您不再在 AWS SCT 项目中使用代理，则可以取消注册。然后，您可以将其注册到其他项目。
4. 选择 Redshift 数据代理，然后选择确定。
5. 请在对话框的连接选项卡上输入您的信息：

   a. 对于描述，请输入代理的描述。
   b. 对于主机名称，请输入代理计算机的主机名称或 IP 地址。
   c. 对于端口，请输入代理将要侦听的端口号。
   d. 选择 “注册” 以在您的 AWS SCT 项目中注册代理。
6. 重复上述步骤，向您的 AWS SCT 项目注册多个代理。

隐藏和恢复 AWS SCT 代理的信息

AWS SCT 代理会加密大量信息，例如用户密钥信任存储的密码、数据库帐户、 AWS 帐户信息以及类似项目。它使用一个名为 `seed.dat` 的特殊文件来进行加密。默认情况下，该代理将在首先配置该代理的用户的工作文件夹中创建此文件。

由于不同的用户可以配置和运行该代理，因此指向 `seed.dat` 的路径存储在 `{extractor.private.folder}` 文件的 `settings.properties` 参数中。当该代理启动时，它可以使用此路径来查找 `seed.dat` 文件以访问要操作的数据库的密钥-信任存储信息。

在下列情况下，您可能需要恢复代理已存储的密码：

- 如果用户丢失了`seed.dat`文件并且 AWS SCT 代理的位置和端口没有改变。
- 如果用户丢失`seed.dat`文件并且 AWS SCT 代理的位置和端口已更改。在这种情况下，变化通常是由于该代理迁移到其他主机或端口而导致的，`seed.dat` 文件中的信息将不再有效。

在这些情况下，如果代理在启动时没有使用 SSL，则它会启动并访问以前创建的代理存储。然后，它会进入 Waiting for recovery (等待恢复) 状态。

但是，在这些情况下，如果代理在启动时使用了 SSL，则无法重新启动它。这是因为代理无法解密存储在 `settings.properties` 文件中的证书的密码。对于这种类型的启动，该代理将无法启动。在日志中将会写入类似下面这样的错误：“The agent could not start with SSL mode enabled。Please reconfigure the agent。Reason: The password for keystore is incorrect.”(代理无法在启用 SSL 模式的情况下启动。请重新配置代理。原因: 密钥存储中的密钥不正确。)

要修复此问题，请创建一个新代理并将其配置为使用现有密码来访问 SSL 证书。为此，请使用以下过程。

执行此过程后，代理应运行并进入 “等待恢复” 状态。 AWS SCT 自动将所需的密码发送给处于 “等待恢复” 状态的代理。当代理获得密码之后，它将重新启动所有任务。 AWS SCT 端不需要任何进一步的用户操作。

重新配置代理并还原用于访问 SSL 证书的密码

1. 安装新的 AWS SCT 代理并运行配置。
2. 将 `agent.name` 文件中的 `instance.properties` 属性更改为创建的存储所用于的代理的名称，以便将新的代理用于现有的代理存储。

   `instance.properties` 文件存储在代理的私有文件夹中，该文件夹使用以下命名约定：{*`output.folder`*}\dmt\{*`hostName`*}_{*`portNumber`*}\。
3. 将 {*`output.folder`*} 的名称更改为以前的代理的输出文件夹的名称。

   此时， AWS SCT 仍在尝试访问旧主机和端口上的旧提取器。结果，无法访问的提取器将获得“失败”状态。然后，您可以更改主机和端口。
4. 通过使用 Modify 命令来修改旧代理的主机或/和端口，从而将请求流重定向到新代理。

当 AWS SCT 能 ping 新代理时， AWS SCT 会收到等待代理恢复的状态。 AWS SCT 然后自动恢复代理的密码。

使用代理存储的每个代理将更新一个名为 `storage.lck` 的、位于 {*`output.folder`*}\{*`agentName`*}\storage\ 的特殊文件。该文件包含代理的网络 ID 和存储锁定的结束时间。当代理使用代理存储时，它每 5 分钟便会更新 `storage.lck` 文件并将存储的租约延长 10 分钟。在租约过期之前，任何其他实例将无法使用此代理存储。

在中创建数据迁移规则 AWS SCT

在使用提取数据之前 AWS Schema Conversion Tool，您可以设置筛选器来减少提取的数据量。您可以使用 WHERE 子句创建数据迁移前规则，以减少提取的数据量。例如，您可以编写一个 WHERE 子句，以从单个表中选择数据。

您可以创建数据迁移规则，并将筛选条件另存为项目的一部分。打开项目，使用以下过程创建数据迁移规则。

创建数据迁移规则

1. 打开视图菜单，然后选择数据迁移视图（其他）。
2. 选择数据迁移规则，然后选择添加新规则。
3. 配置数据迁移规则：

    a. 对于名称，请为数据迁移规则输入一个名称。
    b. 对于架构名称类似于，请输入要应用于架构的筛选条件。在此筛选器中，通过使用 WHERE 子句对 LIKE 子句进行评估。要选择一个架构，请输入确切的架构名称。要选择多个架构，请使用“%”字符作为通配符来匹配架构名称中任意数量的字符。
    c. 对于表名称类似于，请输入要应用于表的筛选条件。在此筛选器中，通过使用 WHERE 子句对 LIKE 子句进行评估。要选择一张表，请输入一个确切的名称。要选择多个表，请使用“%”字符作为通配符来匹配表名中任意数量的字符。
    d. 对于 Where 子句，请输入一个 WHERE 子句以筛选数据。
4. 配置完筛选器后，请选择 Save 以保存您的筛选器，或选择 Cancel 以取消您的更改。
5. 添加、编辑和删除筛选条件后，选择全部保存以保存您的所有更改。

要关闭筛选器而不删除它，请使用“切换”图标。要复制现有的筛选器，请使用“复制”图标。要删除现有筛选器，请使用“删除”图标。要保存对筛选条件所做的所有更改，请选择全部保存。

从项目设置中更改提取器和复制设置

在的 “项目设置” 窗口中 AWS SCT，您可以为数据提取代理和 Amazon Redshift COPY 命令选择设置。

要选择这些设置，请选择设置、项目设置，然后选择数据迁移。在这里，您可以编辑提取设置、Amazon S3 设置和复制设置。

按照下表中的说明提供有关提取设置的信息。

对于此参数	请执行该操作
压缩格式	指定输入文件的压缩格式。选择以下选项之一：GZIP、BZIP2、ZSTD 或不压缩。
分隔符字符	指定用于分隔输入文件中字段的 ASCII 字符。不支持非打印字符。
NULL 值作为字符串	如果您的数据包含 null 终止符，请启用此选项。如果关闭此选项，Amazon Redshift COPY 命令会将空值视为记录结束并结束加载过程。
排序策略	使用排序从故障点重新开始提取。选择以下排序策略之一：在第一次失败后使用排序（推荐）、尽可能使用排序或绝不使用排序。有关更多信息，请参阅 the section called “对数据进行排序”。
源临时架构	输入源数据库中架构的名称，其中提取代理可以创建临时对象。
输出文件大小（MB）	输入已上传到 Amazon S3 的文件大小（MB）。
Snowball 输出文件大小（MB）	输入上传到的文件的大小（以 MB 为单位） AWS Snowball。文件的大小可以是 1-1000 MB。
使用自动分区。对于 Greenplum 和 Netezza，请输入支持的表格的最小大小（兆字节）	启用此选项以使用表分区，然后输入 Greenplum 和 Netezza 源数据库要分区的表的大小。对于 Oracle 到 Amazon Redshift 的迁移，您可以将此字段保留为空，因为 AWS SCT 会为所有分区表创建子任务。
提取 LOB	启用此选项可从源数据库中提取大型对象（LOB）。LOB 包括 BLOB、CLOB、NCLOB、XML 文件等。对于每个 LOB， AWS SCT 提取代理都会创建一个数据文件。
Amazon S3 存储桶 LOB 文件夹	输入 AWS SCT 提取代理存储 LOB 的位置。
将 RTRIM 应用于字符串列	启用此选项可从提取的字符串的末尾剪裁指定的一组字符。

对于此参数	请执行该操作
将文件上传到 Amazon S3 后将文件保存在本地	启用此选项可在数据提取代理将文件上传到 Amazon S3 后将其保留在本地计算机上。

按照下表中的说明提供 Amazon S3 设置的信息。

对于此参数	请执行该操作
使用代理	启用此选项可使用代理服务器将数据上传到 Amazon S3。然后选择数据传输协议，输入主机名称、端口、用户名称和密码。
端点类型	选择 FIPS 以使用美国联邦信息处理标准（FIPS）端点 选择 VPCE 以使用虚拟私有云（VPC）端点。然后，对于 VPC 端点，输入 VPC 端点的域名系统（DNS）。
将文件复制到 Amazon Redshift 后将其保存在 Amazon S3 上	启用此选项可在将提取的文件复制到 Amazon Redshift 后将其保存在 Amazon S3 上。

按照下表中的说明提供复制设置信息。

对于此参数	请执行该操作
最大错误计数	输入加载错误的数量。操作达到此限制后， AWS SCT 数据提取代理将结束数据加载过程。默认值为 0，这意味着无论出现何种故障， AWS SCT 数据提取代理都会继续加载数据。
替换无效的 UTF-8 字符	启用此选项可将无效的 UTF-8 字符替换为指定字符并继续数据加载操作。
空白作为空值	启用此选项可将由空格字符组成的空白字段加载为空。
空作为空值	启用此选项可将空 CHAR 字段和 VARCHAR 字段加载为 null。
截断列	启用此选项可截断列中的数据以符合数据类型规范。

对于此参数	请执行该操作
自动压缩	启用此选项可在复制操作期间应用压缩编码。
自动刷新统计数据	启用此选项可在复制操作结束时刷新统计数据。
加载前检查文件	启用此选项可在将数据文件加载到 Amazon Redshift 之前对其进行验证。

在迁移之前使用对数据进行排序 AWS SCT

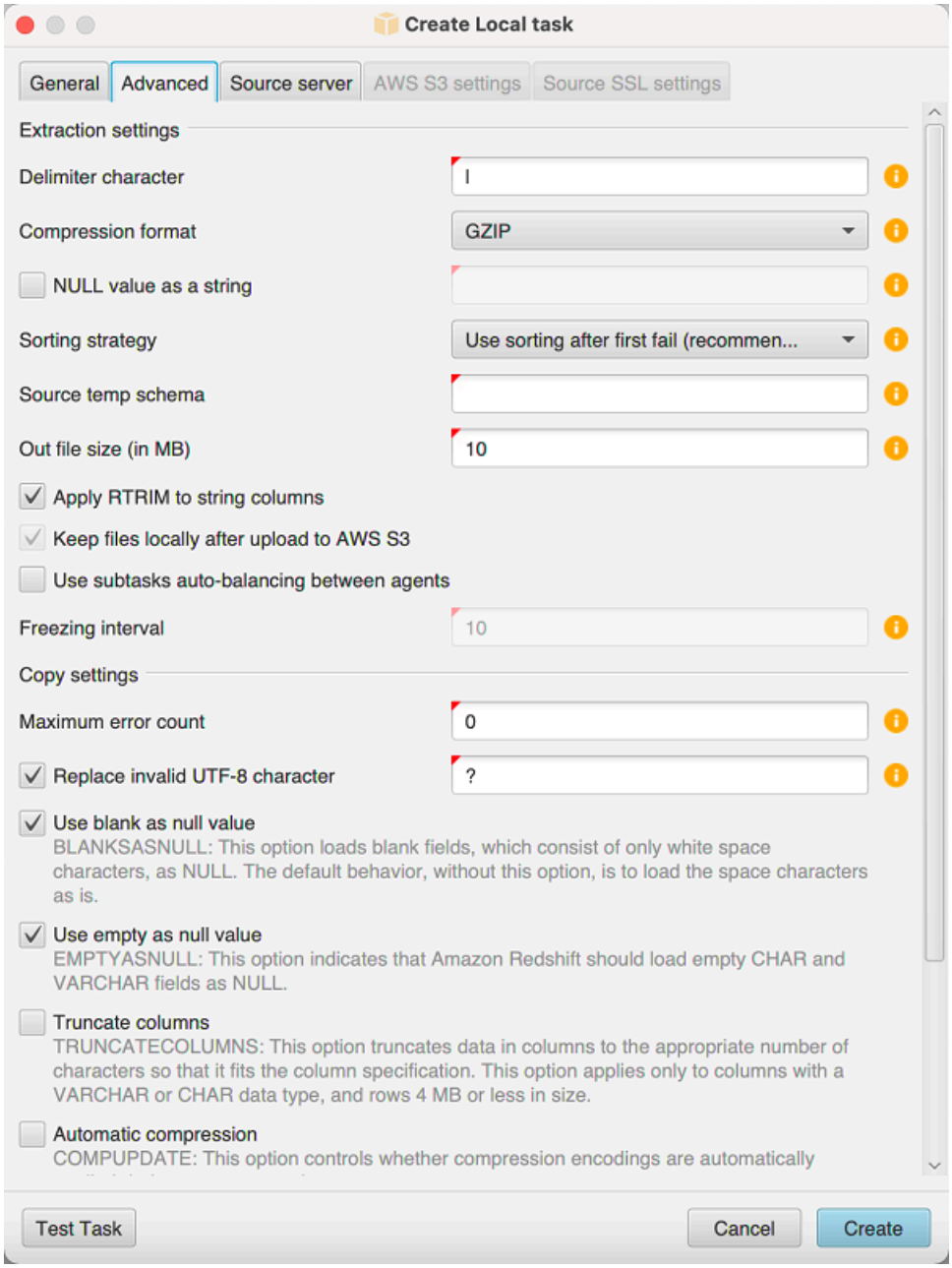
在迁移之前使用对数据进行排 AWS SCT 序有一些好处。如果先对数据进行排序，则 AWS SCT 可以在失败后在最后保存的位置重新启动提取代理。此外，如果您要将数据迁移到 Amazon Redshift 并先对数据进行排序，则 AWS SCT 可以更快地将数据插入到 Amazon Redshift 中。

这些好处与如何 AWS SCT 创建数据提取查询有关。在某些情况下，会在这些查询中 AWS SCT 使用 DENSE_RANK 分析函数。但是，DENSE_RANK 会使用大量时间和服务器资源对提取结果的数据集进行排序，因此，如果没有它 AWS SCT 可以工作，它就可以了。

要在迁移之前使用对数据进行排序 AWS SCT

1. 打开一个 AWS SCT 项目。
2. 打开该对象的上下文 (右键单击) 菜单，然后选择创建本地任务。
3. 选择高级选项卡，然后，对于排序策略，选择选项：

    - 绝不使用排序：提取代理不会使用 DENSE_RANK 分析功能，且如果发生故障，它将重新启动。
    - 尽可能使用排序：如果表有主键或唯一约束，提取代理将使用 DENSE_RANK。
    - 在首次失败后使用排序（推荐）：提取代理首先尝试在不使用 DENSE_RANK 的情况下获取数据。如果首次尝试失败，提取代理将使用 DENSE_RANK 重新构建查询并在发生故障时保留查询的位置。



4. 按下面所示设置其他参数，然后选择 Create 以创建您的数据提取任务。

创建、运行和监控 AWS SCT 数据提取任务

使用以下过程创建、运行和监控数据提取任务。

将任务分配给代理并迁移数据

1. 在中 AWS Schema Conversion Tool，转换架构后，从项目的左侧面板中选择一个或多个表。

   您可以选择所有表，但我们不建议这样做 (出于性能考虑)。我们建议您根据数据仓库中的表大小，为多个表创建多个任务。

2. 打开每个表的上下文 (右键单击) 菜单，然后选择创建任务。此时显示创建本地任务对话框。
3. 对于任务名称，输入任务的名称。
4. 对于迁移模式，请选择下列选项之一：

   - 仅提取：提取您的数据，并将数据保存到您的本地工作文件夹。
   - 提取并上传：提取您的数据，并将数据上传到 Amazon S3。
   - 提取、上传和复制：提取您的数据，将数据上传到 Amazon S3，并将其复制到您的 Amazon Redshift 数据仓库。

5. 对于加密类型，选择下列选项之一：

   - 无：在整个数据迁移过程中关闭数据加密。
   - CSE_SK：使用带有对称密钥的客户端加密来迁移数据。 AWS SCT 使用安全套接字层（SSL）自动生成加密密钥并将其传输到数据提取代理。 AWS SCT 在数据迁移期间不加密大型对象（LOB）。

6. 选择 Extract LOBs 以提取大型对象。如果您不需要提取大型对象，则可以清除该复选框。这样做可以减少提取的数据量。
7. 如果要查看有关任务的详细信息，请选择启用任务日志记录。您可以使用任务日志来调试问题。

   如果启用任务日志记录，请选择要查看的详细信息级别。级别如下所示，每个级别包含上一级别的所有消息：

   - ERROR：最小数量的详细信息。
   - WARNING
   - INFO
   - DEBUG
   - TRACE：最大数量的详细信息。

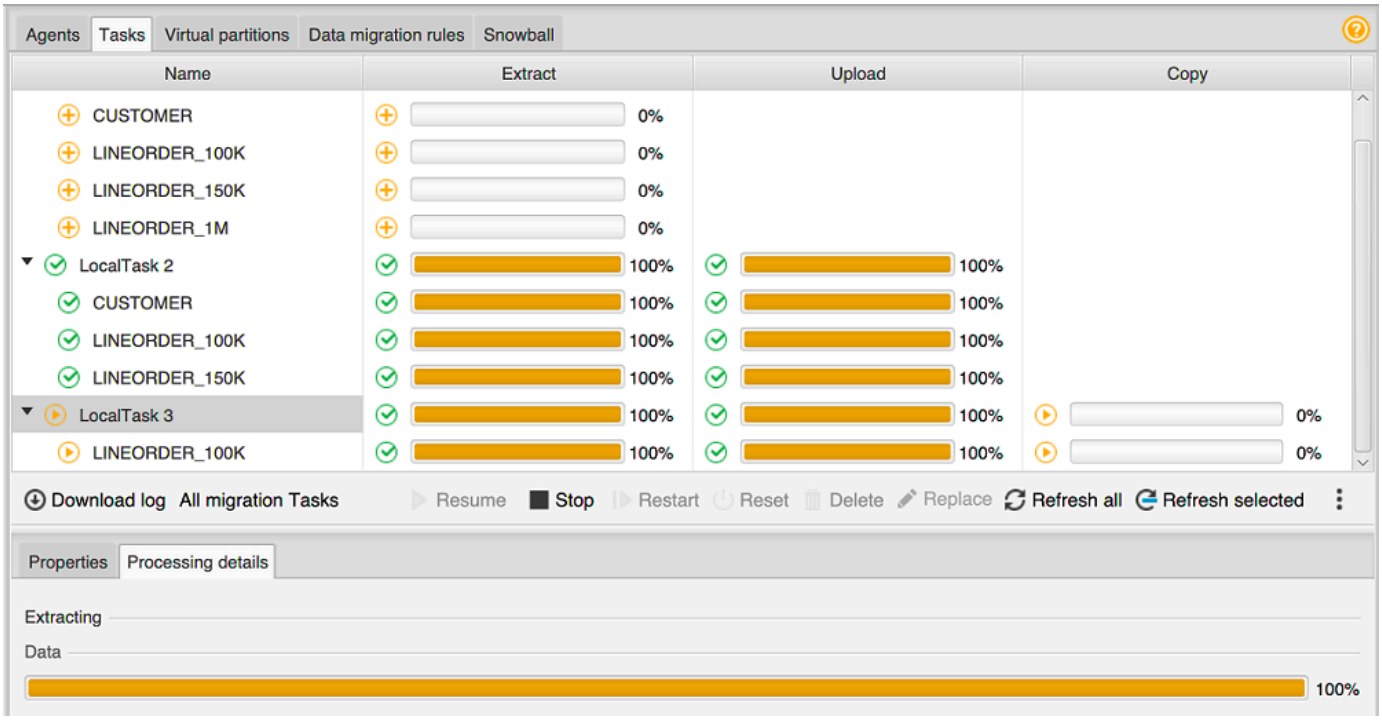
8. 要从中导出数据 BigQuery，请 AWS SCT 使用 Google 云存储空间存储分区文件夹。数据提取代理在此文件夹中存储您的源数据。

要输入 Google Cloud Storage 存储桶文件夹的路径，请选择高级。对于 Google CS 存储桶文件夹，输入存储桶名称和文件夹名称。

9. 要为您的数据提取代理用户代入角色，请选择 Amazon S3 设置。对于 IAM 角色，输入要使用的角色的名称。在 “区域” 中， AWS 区域 为该角色选择。
10. 选择测试任务以验证是否可连接到您的工作文件夹、Amazon S3 存储桶和 Amazon Redshift 数据仓库。验证取决于您选择的迁移模式。
11. 选择 Create (创建) 以创建任务。
12. 重复之前步骤，为您要迁移的所有数据创建任务。

运行和监控任务

1. 对于视图，选择数据迁移视图。此时显示 Agents 选项卡。
2. 选择 Tasks 选项卡。您的任务将显示在顶部网格中，如下所示。您可以在顶部网格中查看任务的状态，并在底部网格中查看其子任务的状态。



3. 在顶部网格中选择一个任务并将其展开。根据所选的迁移模式，您将看到该任务分为 Extract、Upload 和 Copy。
4. 对任务选择 Start 以开始该任务。您可以监控任务工作时的状态。其子任务并行运行。提取、上传和复制操作也并行运行。
5. 如果启用了日志记录，当您设置任务时，可以查看日志：

a. 选择下载日志。此时显示一条消息，其中包含存储该日志文件的文件夹的名称。关闭该消息。

b. Task details 选项卡中显示一个链接。选择该链接以打开包含日志文件的文件夹。

您可以关闭 AWS SCT，您的代理和任务将继续运行。您可以 AWS SCT 稍后重新打开以检查任务状态并查看任务日志。

您可以将数据提取任务保存到本地磁盘，然后使用导出和导入将其还原到同一项目或其他项目。如需导出任务，请确保项目中至少创建了一个提取任务。您可以导入项目中创建的单个提取任务或所有提取任务。

导出提取任务时， AWS SCT 会为该任务创建一个单独的.xml文件。该 .xml 文件存储该任务的元数据信息，例如任务属性、描述和子任务。该 .xml 文件不包含有关提取任务处理的信息。导入任务时，会重新创建如下信息：

- 任务进度
- 子任务和阶段状态
- 按子任务和阶段分配提取代理
- 任务和子任务 ID
- 任务名称

导出和导入 AWS SCT 数据提取任务

您可以快速保存一个项目中的现有任务，然后使用导 AWS SCT 出和导入将其恢复到另一个项目（或同一个项目）中。使用以下过程可导出和导入数据提取任务。

导出和导入数据提取任务

1. 对于视图，选择数据迁移视图。此时显示 Agents 选项卡。
2. 选择 Tasks 选项卡。您的任务将在出现的网格中列出。
3. 选择位于任务列表右下角的三个垂直对齐的点（省略号图标）。
4. 从弹出式菜单中选择导出任务。
5. 选择 AWS SCT 要放置任务导出.xml文件的文件夹。

   AWS SCT 创建文件名格式为的任务导出文件*TASK-DESCRIPTION_TASK-ID*.xml。
6. 选择任务列表右下角的三个垂直对齐的点（省略号图标）。

7. 从弹出式菜单中选择导入任务。

   您可以将提取任务导入到与源数据库相连的项目，并且该项目至少有一个已注册的提取代理处于活动状态。

8. 为导出的提取任务选择 `.xml` 文件。

   AWS SCT 从文件中获取提取任务的参数，创建任务，然后将任务添加到提取代理中。

9. 重复这些步骤以导出和导入其他数据提取任务。

在此过程结束时，您的导出和导入已完成，并且您的数据提取任务已准备就绪。

使用 AWS Snowball Edge 设备提取数据

使用 AWS SCT 和 AWS Snowball Edge 的过程分为几个步骤。迁移涉及本地任务，即 AWS SCT 使用数据提取代理将数据移至 AWS Snowball Edge 设备，然后是将数据从 Ed AWS Snowball ge 设备 AWS 复制到 Amazon S3 存储桶的中间操作。该过程完成了将数据从 Amazon S3 存储桶 AWS SCT 加载到 Amazon Redshift。

本概述之后的各节为每项任务提供了 step-by-step 指南。该过程假设您已在专用计算机上 AWS SCT 安装并配置并注册了数据提取代理。

执行以下步骤，使用 AWS Snowball Edge 将数据从本地数据存储迁移到 AWS 数据存储。

1. 使用 AWS Snowball 控制台创建 AWS Snowball Edge 作业。
2. 使用本地专用 Linux 计算机解锁 AWS Snowball Edge 设备。
3. 在中创建新项目 AWS SCT。
4. 安装和配置数据提取代理。
5. 创建要使用的 Amazon S3 存储桶并设置权限。
6. 将 AWS Snowball 任务导入您的 AWS SCT 项目。
7. 在 AWS SCT中注册数据提取代理。
8. 在中创建本地任务 AWS SCT。
9. 在 AWS SCT中运行和监控数据迁移任务。

S 使用 AWS SCT 和 AWS Snowball Edge 迁移数据的tep-by-step 程序

以下部分提供了有关迁移步骤的详细信息。

步骤 1：创建 AWS Snowball Edge 作业

按照《AWS Snowball 边AWS Snowball 缘开发者指南》中创建 Edge AWS Snowball Job 一节中概述的步骤创建作业。

第 2 步：解锁 AWS Snowball Edge 设备

运行命令，从安装代理的计算机上解锁 Snowball Edge 设备并向 Snowball Edge 设备提供凭据。 AWS DMS 通过运行这些命令，可以确保 AWS DMS 代理呼叫连接到 AWS Snowball Edge 设备。有关解锁 AWS Snowball Edge 设备的更多信息，请参阅解锁 Snowball Edge。

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

第 3 步：创建新 AWS SCT 项目

接下来，创建一个新 AWS SCT 项目。

要在中创建新项目 AWS SCT

1. 启动 AWS Schema Conversion Tool. 在文件菜单上，选择新建项目。此时将显示新建项目对话框。
2. 为本地存储在计算机上的项目输入一个名称。
3. 输入本地项目文件的位置。
4. 选择 “确定” 创建您的 AWS SCT 项目。
5. 选择 “添加源”，将新的源数据库添加到您的 AWS SCT 项目中。
6. 选择添加目标可在 AWS SCT 项目中添加新的目标平台。
7. 在左侧面板中选择源数据库架构。
8. 在右侧面板中，为所选源架构指定目标数据库平台。
9. 选择创建映射。选择源数据库架构和目标数据库平台后，此按钮将变为活动状态。

步骤 4：安装和配置数据提取代理

AWS SCT 使用数据提取代理将数据迁移到 Amazon Redshift。您下载用于安装的.zip 文件 AWS SCT 包括解压缩代理安装程序文件。您可以在 Windows、Red Hat Enterprise Linux 或 Ubuntu 中安装数据提取代理。有关更多信息，请参阅 安装提取代理。

要配置数据提取代理，请输入源数据库引擎和目标数据库引擎。此外，请确保在运行数据提取代理的计算机上下载了源数据库和目标数据库的 JDBC 驱动程序。数据提取代理使用这些驱动程序连接至源数据库和目标数据库。有关更多信息，请参阅 下载所需的数据库驱动程序。

在 Windows 中，数据提取代理安装程序在命令提示符窗口中启动配置向导。在 Linux 中，从安装代理的位置运行该 `sct-extractor-setup.sh` 文件。

步骤 5：配置 AWS SCT 为访问 Amazon S3 存储桶

有关 Amazon S3 存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的存储桶概述。

第 6 步：将 AWS Snowball 作业导入到您的 AWS SCT 项目中

要将您的 AWS SCT 项目与 AWS Snowball Edge 设备连接，请导入您的 AWS Snowball 作业。

导入您的 AWS Snowball 作业

1. 打开设置菜单，然后选择全局设置。此时显示 Global settings 对话框。
2. 选择 AWS 服务配置文件，然后选择导入作业。
3. 选择你的 AWS Snowball 工作。
4. 输入 AWS Snowball IP。有关更多信息，请参阅《AWS Snowball 用户指南》中的更改 IP 地址。
5. 输入您的 AWS Snowball 端口。有关更多信息，请参阅《AWS Snowball 边AWS Snowball 缘开发者指南》中的在 Edge 设备上使用 AWS 服务所需的端口。
6. 输入您的 AWS Snowball 访问密钥和 AWS Snowball 秘密密钥。有关更多信息，请参阅《AWS Snowball 用户指南》中的 AWS Snowball中的身份验证和访问控制。
7. 选择 Apply，然后选择 OK。

步骤 7：在中注册数据提取代理 AWS SCT

在本节中，您将在 AWS SCT中注册数据提取代理。

注册数据提取代理

1. 在视图菜单上，选择数据迁移视图（其他），然后选择注册。
2. 对于描述，请输入数据提取代理名称。
3. 对于主机名，请输入运行数据提取代理的计算机的 IP 地址。
4. 对于端口，请输入您配置的侦听端口。

5. 选择 Register。

步骤 8：创建本地任务

接下来，创建迁移任务。该任务包含两个子任务。一个子任务将数据从源数据库迁移到 AWS Snowball Edge 设备。另一个子任务提取该设备加载到 Amazon S3 存储桶的数据，并将其迁移到目标数据库。

创建迁移任务

1. 打开视图菜单，然后选择数据迁移视图（其他）。
2. 在显示源数据库架构的左侧面板中，选择一个要迁移的架构对象。打开该对象的上下文 (右键单击) 菜单，然后选择创建本地任务。
3. 对于任务名称，请输入数据迁移任务的描述性名称。
4. 对于迁移模式，选择提取、上传和复制。
5. 选择 Amazon S3 设置。
6. 选择使用 Snowball。
7. 在 Amazon S3 存储桶中输入数据提取代理可以存储数据的文件夹和子文件夹。
8. 选择 Create (创建) 以创建任务。

步骤 9：在中运行和监控数据迁移任务 AWS SCT

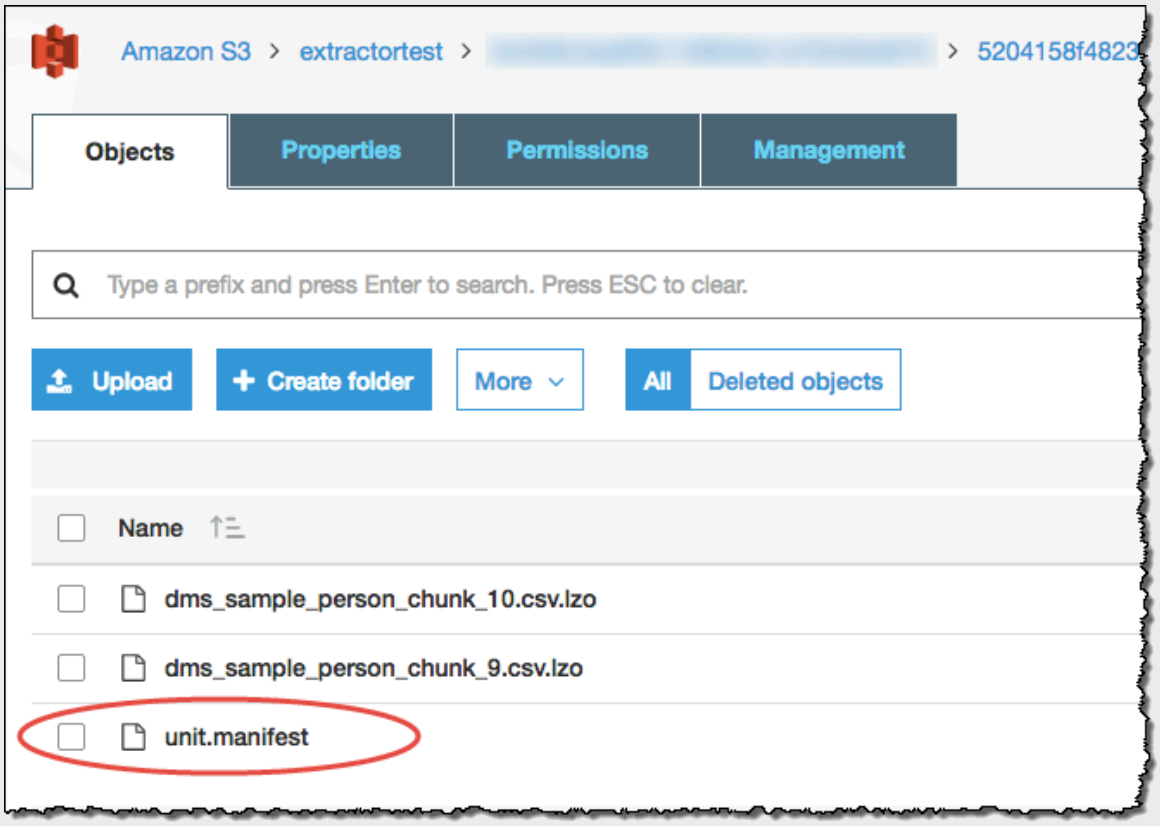
要启动数据迁移任务，请选择开始。确保在上建立了与源数据库、Amazon S3 存储桶、 AWS Snowball 设备的连接以及与目标数据库的连接 AWS。

您可以在任务选项卡中监控和管理数据迁移任务及其子任务。您可以查看数据迁移进度，也可以暂停或重启数据迁移任务。

数据提取任务输出

迁移任务完成后，您的数据便准备就绪。使用以下信息确定如何根据所选的迁移模式和数据位置继续操作。

迁移模式	数据位置
提取、上传和复制	数据已在您的 Amazon Redshift 数据仓库中。您可以验证数据是否存在，并开始使用它。有关更多信息，请参阅通过客户端工具和代码连接到集群。

迁移模式	数据位置
提取和上传	提取代理已将您的数据以文件形式保存在 Amazon S3 存储桶中。您可以使用 Amazon Redshift 复制命令将数据加载到 Amazon Redshift。有关更多信息，请参阅 Amazon Redshift 文档中的从 Amazon S3 加载数据。 您的 Amazon S3 存储桶中包含多个文件夹，与您设置的提取任务对应。将数据加载到 Amazon Redshift 时，请指定每个任务所创建的清单文件的名称。清单文件显示在 Amazon S3 存储桶的任务文件夹中，如下所示。 
仅提取	提取代理已将您的数据以文件形式保存在工作文件夹中。手动将数据复制到 Amazon S3 存储桶，然后按照提取和上传的说明继续操作。

将虚拟分区与 AWS Schema Conversion Tool

通常，您可以通过创建子任务 (使用筛选规则创建表数据的虚拟分区) 来最有效地管理大型未分区表。在中 AWS SCT，您可以为迁移的数据创建虚拟分区。有三种可与特定数据类型搭配使用的分区类型：

- RANGE 分区类型，可与数字以及日期和时间数据类型搭配使用。

- LIST 分区类型，可与数字、字符以及日期和时间数据类型搭配使用。
- DATE AUTO SPLIT 分区类型，可与数字、日期和时间数据类型搭配使用。

AWS SCT 验证您为创建分区提供的值。例如，如果您尝试对数据类型为 NUMERIC 的列进行分区，但却提供了不同的数据类型的值，则 AWS SCT 会引发错误。

此外，如果您使用将数据迁移 AWS SCT 到 Amazon Redshift，则可以使用本机分区来管理大型表的迁移。有关更多信息，请参阅 使用本机分区。

创建虚拟分区时的限制

以下是创建虚拟分区的限制：

- 您只能对未分区的表使用虚拟分区。
- 您只能在数据迁移视图中使用虚拟分区。
- 您无法对虚拟分区使用选项 UNION ALL VIEW。

RANGE 分区类型

RANGE 分区类型基于针对数字以及日期和时间数据类型的一系列值对数据进行分区。此分区类型将创建一个 WHERE 子句，您将为每个分区提供值范围。使用值框为已分区的列指定一个值列表。您可以使用 .csv 文件加载值信息。

RANGE 分区类型在分区值的两端创建默认分区。这些默认分区会捕获任何小于或大于指定分区值的数据。

例如，您可以基于您提供的值范围创建多个分区。在以下示例中，指定了 LO_TAX 的分区值以创建多个分区。

```
Partition1: WHERE LO_TAX <= 10000.9
Partition2: WHERE LO_TAX > 10000.9 AND LO_TAX <= 15005.5
Partition3: WHERE LO_TAX > 15005.5 AND LO_TAX <= 25005.95
```

创建 RANGE 虚拟分区

1. 打开 AWS SCT。
2. 选择数据迁移视图（其他）模式。

3. 选择要在其中设置虚拟分区的表。打开该表的上下文 (右键单击) 菜单，并选择添加虚拟分区。
4. 在添加虚拟分区对话框中，输入如下信息。

选项	操作
分区类型	选择 RANGE。根据您选择的类型，对话框 UI 将发生更改。
列名称	选择要分区的列。
列类型	在该列中选择值的数据类型。
值	通过在 New Value 框中键入各个值来添加新值，然后选择加号以添加相应值。
从文件加载	（可选）输入包含分区值的 .csv 文件的名称。

5. 选择确定。

LIST 分区类型

LIST 分区类型基于一个数字、字符以及日期和时间数据类型的列值对数据进行分区。此分区类型将创建一个 WHERE 子句，您将为每个分区提供值。使用值框为已分区的列指定一个值列表。您可以使用 .csv 文件加载值信息。

例如，您可以基于您提供的值创建多个分区。在以下示例中，指定了 LO_ORDERKEY 的分区值以创建多个分区。

```
Partition1: WHERE LO_ORDERKEY = 1
Partition2: WHERE LO_ORDERKEY = 2
Partition3: WHERE LO_ORDERKEY = 3
…
PartitionN: WHERE LO_ORDERKEY = USER_VALUE_N
```

您还可以为未包含在指定的分区中的值创建默认分区。

如果要从迁移中排除特定值，则可以使用 LIST 分区类型来筛选源数据。例如，假设您要省略带 LO_ORDERKEY = 4 的行。在这种情况下，请不要在分区值列表中包含该值 4，并确保未选择包括其他值。

创建 LIST 虚拟分区

1. 打开 AWS SCT。
2. 选择数据迁移视图（其他）模式。
3. 选择要在其中设置虚拟分区的表。打开该表的上下文 (右键单击) 菜单，并选择添加虚拟分区。
4. 在添加虚拟分区对话框中，输入如下信息。

选项	操作
分区类型	选择 LIST。根据您选择的类型，对话框 UI 将发生更改。
列名称	选择要分区的列。
新值	在此处键入一个值以将其添加到分区值集。
包括其他值	选择此选项可创建一个默认分区，不满足分区条件的所有值都将存储在该分区中。
从文件加载	（可选）输入包含分区值的 .csv 文件的名称。

5. 选择确定。

DATE AUTO SPLIT 分区类型

DATE AUTO SPLIT 分区类型是一种自动生成 RANGE 分区的方式。使用 DATA AUTO SPLIT，您可以告诉 AWS SCT 分区属性、起点和结束位置以及值之间的范围大小。然后， AWS SCT 自动计算分区值。

DATA AUTO SPLIT 可自动执行创建范围分区所涉及的大量工作。使用这种技术和范围分区之间的权衡是您对分区边界的控制程度。自动拆分过程始终创建大小相等（统一）的范围。范围分区使您可以根据特定数据分布的需要改变每个范围的大小。例如，您可以使用每日、每周、每两周、每月等。

```
Partition1: WHERE LO_ORDERDATE >= ‘1954-10-10’ AND LO_ORDERDATE < ‘1954-10-24’
Partition2: WHERE LO_ORDERDATE >= ‘1954-10-24’ AND LO_ORDERDATE < ‘1954-11-06’
Partition3: WHERE LO_ORDERDATE >= ‘1954-11-06’ AND LO_ORDERDATE < ‘1954-11-20’
…
PartitionN: WHERE LO_ORDERDATE >= USER_VALUE_N AND LO_ORDERDATE <= ‘2017-08-13’
```

创建 DATE AUTO SPLIT 虚拟分区

1. 打开 AWS SCT。
2. 选择数据迁移视图（其他）模式。
3. 选择要在其中设置虚拟分区的表。打开该表的上下文 (右键单击) 菜单，并选择添加虚拟分区。
4. 在添加虚拟分区对话框中，输入如下信息。

选项	操作
分区类型	选择 DATE AUTO SPLIT。根据您选择的类型，对话框 UI 将发生更改。
列名称	选择要分区的列。
开始日期	键入开始日期。
结束日期	键入结束日期。
Interval	输入间隔单位，然后选择该单位的值。

5. 选择 确定。

使用本机分区

为了加快数据迁移速度，您的数据提取代理可以使用源数据仓库服务器上表的本机分区。 AWS SCT 支持原生分区，用于从 Greenplum、Netezza 和 Oracle 迁移到 Amazon Redshift。

例如，在创建项目后，您可以收集架构的统计信息并分析选定要迁移的表的大小。对于超过指定大小的表， AWS SCT 会触发本机分区机制。

使用本机分区

1. 打开 AWS SCT，然后为 “文件” 选择 “新建项目”。此时将显示新建项目对话框。
2. 创建新项目，添加源服务器和目标服务器，并创建映射规则。有关更多信息，请参阅 创建 AWS SCT 项目。
3. 选择视图，然后选择主视图。
4. 在项目设置中，选择数据迁移选项卡。选择使用自动分区。对于 Greenplum 和 Netezza 源数据库，请输入支持表的最小大小，以兆字节为单位（例如 100）。 AWS SCT 自动为每个不为空的

本机分区创建单独的迁移子任务。对于 Oracle 到 Amazon Redshift 的迁移，需要为所有分区表 AWS SCT 创建子任务。

5. 在显示源数据库中的架构的左面板中，选择架构。打开该对象的上下文 (右键单击) 菜单，然后选择收集统计数据。要将数据从 Oracle 迁移到 Amazon Redshift，您可以跳过此步骤。
6. 选择所有要迁移的表。
7. 注册所需数量的代理。有关更多信息，请参阅 在中注册提取剂 AWS Schema Conversion Tool。
8. 为所选表创建数据提取任务。有关更多信息，请参阅 创建、运行和监控 AWS SCT 数据提取任务。

   检查大型表是否被拆分为子任务，以及每个子任务是否与呈现源数据仓库中一个切片上的表的一部分的数据集相匹配。
9. 启动并监控迁移过程，直到 AWS SCT 数据提取代理完成源表中的数据迁移。

将 LOB 迁移到 Amazon Redshift

Amazon Redshift 不支持存储大型二进制对象 (LOB)。但是，如果您需要将一个或多个 LOB 迁移到 Amazon Redshift AWS SCT ，可以执行迁移。为此， AWS SCT 使用 Amazon S3 存储桶来存储 LOB 并将该 Amazon S3 存储桶的 URL 写入到 Amazon Redshift 中存储的已迁移数据。

将 LOB 迁移到 Amazon Redshift

1. 打开一个 AWS SCT 项目。
2. 连接到源数据库和目标数据库。刷新目标数据库中的元数据，并确保转换后的表存在于目标数据库中。
3. 对于操作，选择创建本地任务。
4. 对于迁移模式，请选择下列选项之一：

   - 提取并上传提取您的数据，并将数据上传到 Amazon S3。
   - 提取、上传和复制提取您的数据，将数据上传到 Amazon S3，并将其复制到 Amazon Redshift 数据仓库。
5. 选择Amazon S3 设置。
6. 对于 Amazon S3 存储桶 LOB 文件夹，键入 Amazon S3 存储桶中要存储 LOB 的文件夹的名称。

   如果您使用 AWS 服务配置文件，则此字段为可选字段。 AWS SCT 可以使用您个人资料中的默认设置。要使用其他 Amazon S3 存储桶，请在此处输入路径。

7. 启用使用代理选项，使用代理服务器将数据上传到 Amazon S3。然后选择数据传输协议，输入主机名称、端口、用户名称和密码。
8. 对于加密类型，选择 FIPS 以使用联邦信息处理标准（FIPS）端点。选择 VPCE 以使用虚拟私有云（VPC）端点。然后，对于 VPC 端点，输入 VPC 端点的域名系统（DNS）。
9. 打开文件复制到 Amazon Redshift 后将其保留在 Amazon S3 上选项，以便在将提取的文件复制到 Amazon Redshift 后将这些文件保留在 Amazon S3 上。
10. 选择 Create (创建) 以创建任务。

数据提取代理的最佳实践和故障排除

以下是一些关于使用提取代理的最佳实践和故障排除建议。

问题	故障排除建议
性能低下	为了提高性能，我们建议执行以下操作： • 安装多个代理。 • 在靠近数据仓库的计算机上安装代理。 • 请勿对单个代理任务运行所有表。
争用延迟	避免有太多代理同时访问您的数据仓库。
代理暂时关闭	如果某代理关闭，其每个任务在 AWS SCT中的状态显示为失败。如果等待，某些情况下，该代理可以恢复。在这种情况下，其任务的状态会在 AWS SCT中更新。
代理永久关闭	如果运行代理的计算机永久关闭，而该代理正运行一个任务，您可以替换新的代理以继续该任务。仅当原始代理的工作文件夹与原始代理不位于同一台计算机时，您才可以使用新代理进行替换。要替换新的代理，请执行以下操作： • 在新计算机上安装代理。 • 对新代理进行配置，以便与原始代理具有相同设置，包括端口号和工作文件夹。 • 启动该代理。代理启动后，该任务发现新的可用代理并继续在新代理上运行。

使用 AWS SCT 转换应用程序 SQL

当您将数据库架构从一个引擎转换到另一个引擎时，还需要更新应用程序中的 SQL 代码，以便与新数据库引擎 (而非旧引擎) 进行交互。您可以查看、分析、编辑和保存转换后的 SQL 代码。

您可以使用 AWS Schema Conversion Tool（AWS SCT）通过 C++、C#、Java 或其他应用程序代码转换 SQL 代码。对于 Oracle 到 PostgreSQL 转换，可以使用 AWS SCT 将 SQL*Plus 代码转换为 PSQL。此外，对于 Oracle 到 PostgreSQL 的转换，你可以使用 AWS SCT 转换嵌入到 C#、C++、Java 和 Pro*C 应用程序中的 SQL 代码。

主题

转换应用程序 SQL 概述

要转换应用程序中的 SQL 代码，请执行以下简要步骤：

- 创建一个应用程序转换项目：应用程序转换项目是数据库架构转换项目的子级。每个数据库架构转换项目都可以有一个或多个子应用程序转换项目。有关更多信息，请参阅在 AWS SCT 中创建通用应用程序转换项目。
- 分析和转换 SQL 代码：可以分析您的应用程序，提取 SQL 代码，并创建一个本地版本的转换后 SQL，以便您查看和编辑。在您做好准备之前，该工具不会更改您的应用程序中的代码。有关更多信息，请参阅在 AWS SCT 中分析和转换 SQL 代码。
- 创建应用程序评估报告：应用程序评估报告可提供关于将应用程序 SQL 代码从源数据库架构转换到目标数据库架构的重要信息。有关更多信息，请参阅在 AWS SCT 中创建和使用 AWS SCT 评估报告。。
- 编辑、应用更改并保存转换后的 SQL 代码：评估报告包含无法自动转换的 SQL 代码项的列表。对于这些项目，您可以手动编辑 SQL 代码以执行转换。有关更多信息，请参阅使用 AWS SCT 编辑和保存转换后的 SQL 代码。

使用 AWS SCT 转换应用程序中的 SQL 代码

您可以使用 AWS SCT 转换嵌入到应用程序中的 SQL 代码。通用 AWS SCT 应用程序转换器将您的应用程序代码视为纯文本。它会扫描您的应用程序代码并使用正则表达式提取 SQL 代码。该转换器支持不同类型的源代码文件，并且可以处理用任何编程语言编写的应用程序代码。

通用应用程序转换器具有以下限制。它不会深入探讨特定于应用程序编程语言的应用程序逻辑。此外，通用转换器不支持来自不同应用程序对象（例如函数、参数、局部变量等）的 SQL 语句。

要改进应用程序 SQL 代码转换，请使用特定于语言的应用程序 SQL 代码转换器。有关更多信息，请参阅转换 C# 应用程序中的 SQL 代码、转换 Java 应用程序中的 SQL 代码和转换 Pro*C 应用程序中的 SQL 代码。

在 AWS SCT 中创建通用应用程序转换项目

在 AWS Schema Conversion Tool 中，应用程序转换项目是数据库架构转换项目的子级。每个数据库架构转换项目都可以有一个或多个子应用程序转换项目。

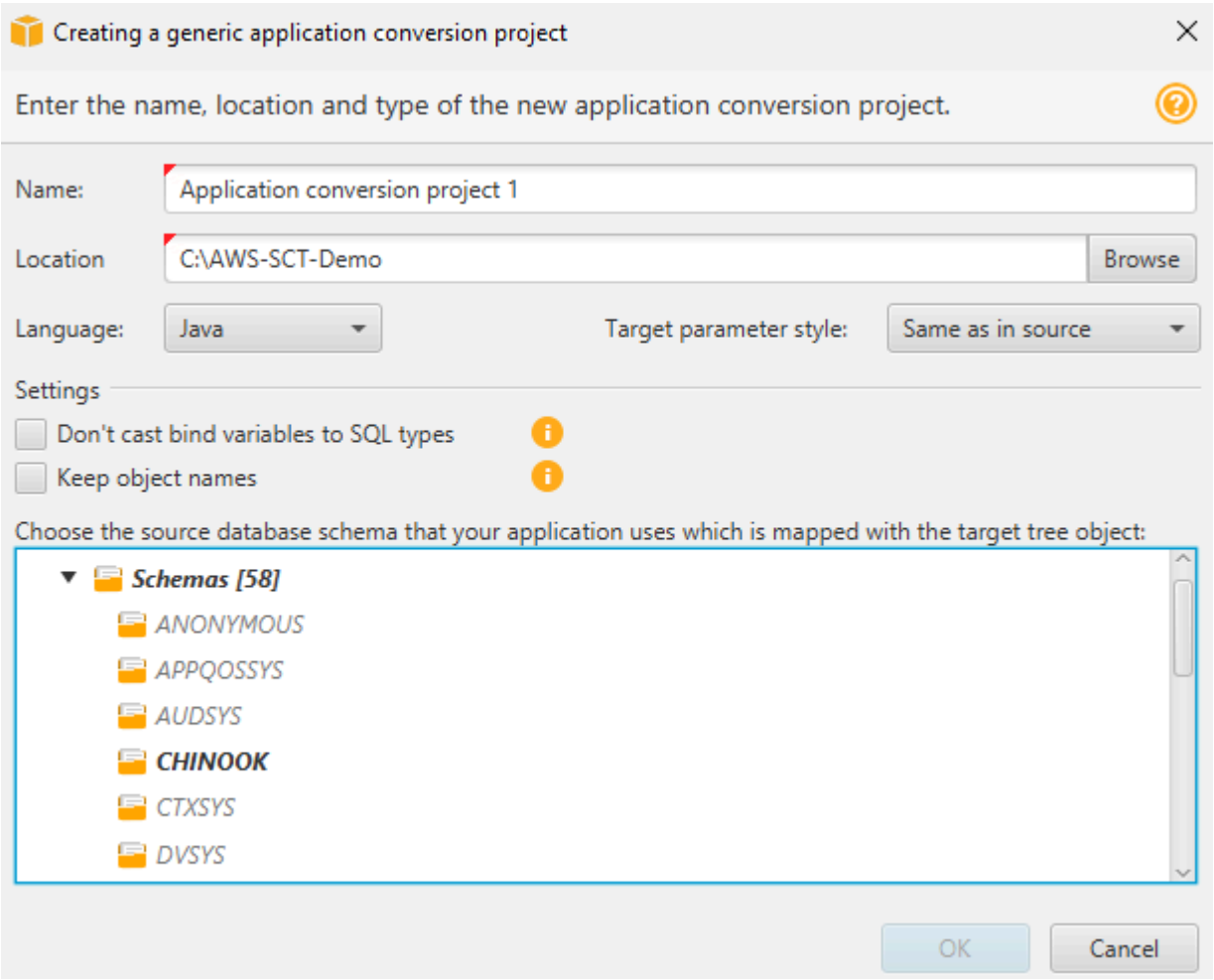
> **Note**
>
> AWS SCT 不支持在以下源和目标之间进行转换：
>
> - Oracle 到 Oracle
> - PostgreSQL 到 PostgreSQL 或 Aurora PostgreSQL
> - MySQL 到 MySQL
> - SQL Server 到 SQL Server
> - Amazon Redshift 到 Amazon Redshift
> - SQL Server 到 Babelfish
> - SQL Server Integration Services 到 AWS Glue
> - Apache Cassandra 到 Amazon DynamoDB

使用以下过程创建通用应用程序转换项目。

创建应用程序转换项目

1. 在 AWS Schema Conversion Tool 中，在应用程序菜单上选择新建通用应用程序。

随即出现 New application conversion project 对话框。



2. 添加以下项目信息。

对于此参数	请执行该操作
名称	输入您的应用程序转换项目的名称。每个数据库架构转换项目都可以有一个或多个子应用程序转换项目，因此如果您在事后添加更多项目的话，请选择一个有意义的名称。
位置	输入您的应用程序源代码的位置。
语言	选择以下选项之一： • Java • C++

对于此参数	请执行该操作
	• C# • 任何
目标参数样式	选择转换后的代码中用于绑定变量的语法。不同的数据库平台对绑定变量使用不同的语法。请选择以下任一选项： • Same as in source • Positional (?) • Indexed (:1) • Indexed ($1) • Named (@name) • Named (:name) • Named (&name) • Named ($name) • Named (#name) • Named (!name!)
选择源数据库架构	在源树状图中，选择应用程序所用的架构。确保此架构是映射规则的一部分。

3. 选择不要将绑定变量转换为 SQL 类型，以避免将绑定变量类型转换为 SQL 类型。该选项仅适用于 Oracle 到 PostgreSQL 的转换。

   例如，您的源应用程序代码包括以下 Oracle 查询：

   ```
   SELECT * FROM ACCOUNT WHERE id = ?
   ```

   如果选择不要将绑定变量转换为 SQL 类型，则 AWS SCT 会按如下所示转换此查询。

   ```
   SELECT * FROM account WHERE id = ?
   ```

   清除不要将绑定变量转换为 SQL 类型后，AWS SCT 会将绑定变量类型更改为 NUMERIC 数据类型。转换结果如下所示。

```
SELECT * FROM account WHERE id = (?)::NUMERIC
```

4. 选择保留对象名称，以避免将架构名称添加到已转换对象的名称中。该选项仅适用于 Oracle 到 PostgreSQL 的转换。

   例如，假定您的源应用程序代码包含以下 Oracle 查询。

   ```
   SELECT * FROM ACCOUNT
   ```

   当您选择保留对象名称时，AWS SCT 将转换此查询，如下所示。

   ```
   SELECT * FROM account
   ```

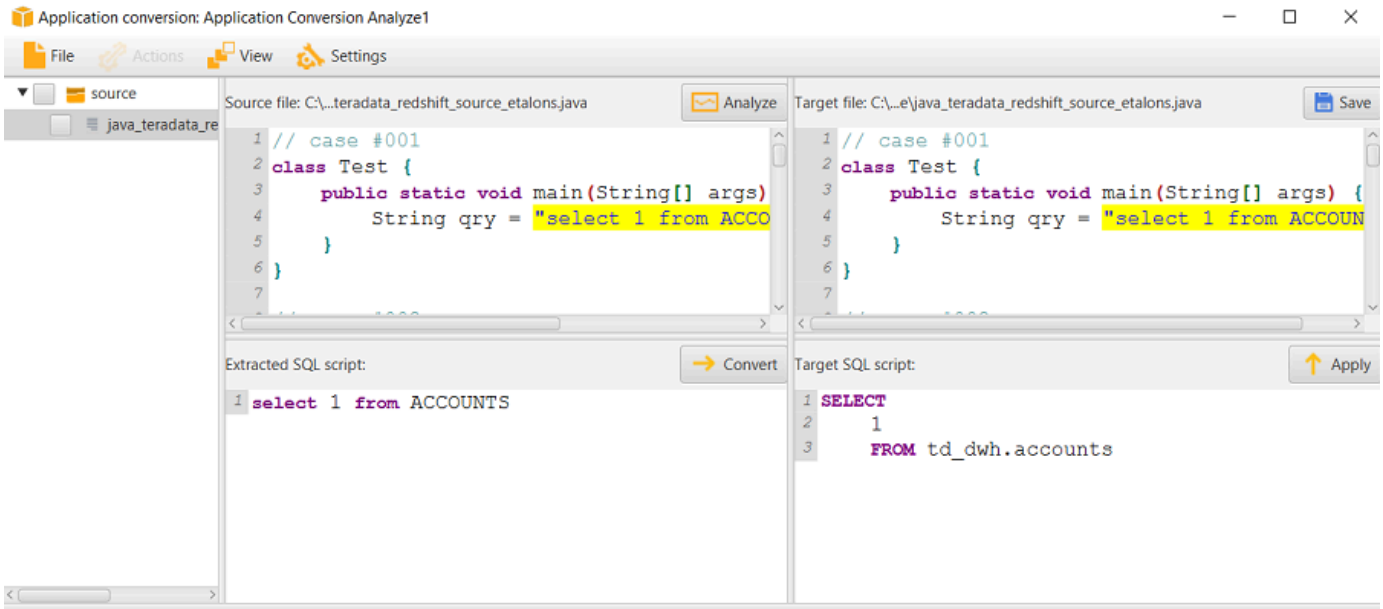
   清除保留对象名称后，AWS SCT 会将架构名称添加到表的名称中。转换结果如下所示。

   ```
   SELECT * FROM schema_name.account
   ```

   如果您的源代码在对象的名称中包含父对象的名称，则 AWS SCT 在转换后的代码中使用此格式。在这种情况下，请忽略保留对象名称选项，因为 AWS SCT 会在转换后的代码中添加父对象的名称。

5. 选择 OK 以创建应用程序转换项目。

   随即打开项目窗口。



在 AWS SCT 中管理应用程序转换项目

您可以打开现有的应用程序转换项目并添加多个应用程序转换项目。

在您创建应用程序转换项目时，项目窗口会自动打开。您可以关闭应用程序转换项目窗口，稍后再回来。

打开现有应用程序转换项目

1. 在左侧面板中，选择应用程序转换项目节点，然后打开上下文（右键单击）菜单。
2. 选择管理应用程序。

添加其他应用程序转换项目

1. 在左侧面板中，选择应用程序转换项目节点，然后打开上下文（右键单击）菜单。
2. 选择 New application（新建应用程序）。
3. 输入创建新的应用程序转换项目所需的信息。有关更多信息，请参阅创建通用应用程序转换项目。

在 AWS SCT 中分析和转换 SQL 代码

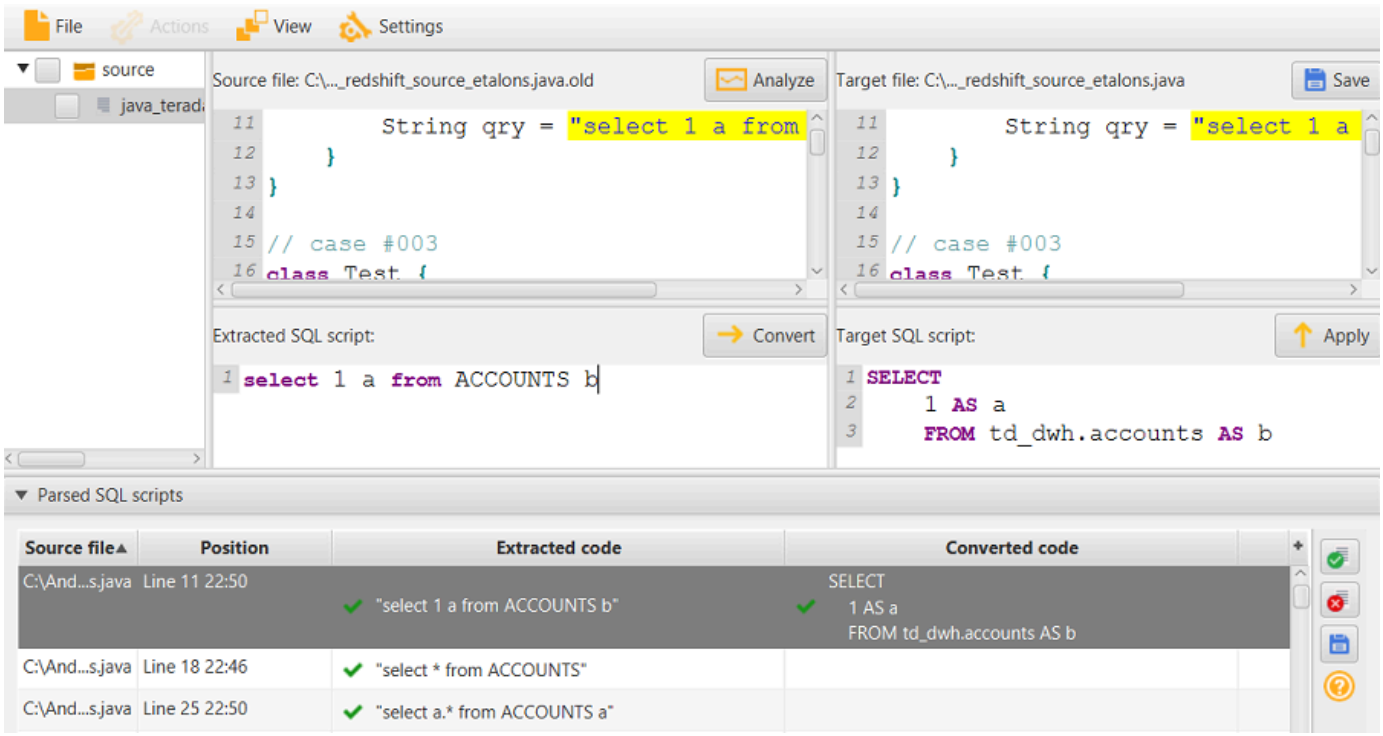
按照以下过程在 AWS Schema Conversion Tool 中分析和转换 SQL 代码。

分析和转换您的 SQL 代码

1. 打开现有的应用程序转换项目，然后选择分析。

   AWS SCT 分析应用程序代码并提取 SQL 代码。AWS SCT 在已解析的 SQL 脚本列表中显示提取的 SQL 代码。
2. 对于已解析的 SQL 脚本，请选择一个项目来查看其提取的 SQL 代码。AWS SCT 在提取的 SQL 脚本窗格中显示所选项目的代码。
3. 选择转换，在提取的 SQL 脚本窗格中转换 SQL 代码。AWS SCT 将代码转换为与目标数据库兼容的格式。

   您可以编辑转换后的 SQL 代码。有关更多信息，请参阅编辑和保存转换后的 SQL 代码。



4. 创建应用程序转换评估报告时，AWS SCT 会转换所有提取的 SQL 代码项。有关更多信息，请参阅创建和使用评估报告。

在 AWS SCT 中创建和使用 AWS SCT 评估报告。

应用程序转换评估报告提供了有关将应用程序 SQL 代码转换为与目标数据库兼容的格式的信息。该报告详细列出了所有提取的 SQL 代码、所有经过转换的 SQL 代码以及无法转换的 SQL 的操作项。

创建应用程序转换评估报告

使用以下过程创建应用程序转换评估报告。

创建应用程序转换评估报告

1. 在应用程序转换项目窗口中，在操作 菜单上选择创建报告。

   AWS SCT 创建应用程序转换评估报告并在应用程序转换项目窗口中将其打开。

2. 查看 Summary 选项卡。

   如下所示的摘要选项卡显示了应用程序评估报告中的摘要信息。它显示了已自动转换的 SQL 代码项目及未自动转换的项目。

3. 选择 SQL 提取操作。

   查看 AWS SCT 无法从源代码中提取的 SQL 代码项列表。

4. 选择 SQL 转换操作。

   查看 AWS SCT 无法自动转换的 SQL 代码项列表。使用推荐的操作手动转换 SQL 代码。有关如何编辑转换的 SQL 代码的信息，请参阅 使用 AWS SCT 编辑和保存转换后的 SQL 代码。

5. （可选）将报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件：

   - 选择右上角的保存为 PDF，将报告另存为 PDF 文件。

     PDF 文件包含应用程序转换的执行摘要、操作项和建议。

   - 选择右上角的保存为 CSV，将报告另存为 CSV 文件。

     CSV 文件包含操作项、推荐的操作以及转换 SQL 代码所需的估计人工操作的复杂性。

使用 AWS SCT 编辑和保存转换后的 SQL 代码

评估报告包含 AWS SCT 无法转换的 SQL 代码项的列表。对于每个项目，AWS SCT 在 SQL 转换操作选项卡上创建一个操作项。对于这些项目，您可以手动编辑 SQL 代码以执行转换。

按照以下过程编辑转换后的 SQL 代码、应用更改，然后进行保存。

编辑、应用更改并保存转换后的 SQL 代码

1. 直接在 Target SQL script 窗格中编辑转换后的 SQL 代码。如果未显示转换后的代码，您可以单击窗格并开始键入。
2. 编辑完转换后的 SQL 代码，请选择 Apply。此时，更改便会保存在内存中，但尚未写入您的文件。
3. 选择 Save 将更改保存到文件中。

   当您选择保存时，会覆盖您的原始文件。在保存之前复制您的原始文件，以便记录原始应用程序代码。

使用 AWS SCT 转换 C# 应用程序中的 SQL 代码

对于 Oracle 到 PostgreSQL 的转换，可以使用 AWS Schema Conversion Tool （AWS SCT）转换嵌入到 C# 应用程序中的 SQL 代码。这个特定的 C# 应用程序转换器了解应用程序逻辑。它收集位于不同应用程序对象中的语句，例如函数、参数、局部变量等。

由于这种深入的分析，C# 应用程序 SQL 代码转换器比通用转换器的转换结果更好。

在 AWS SCT 中创建 C# 应用程序转换项目

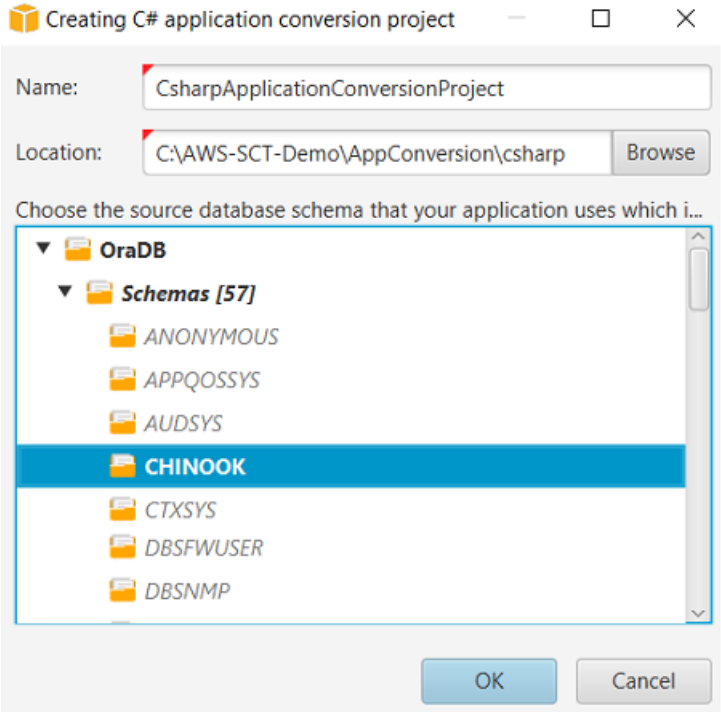
您可以仅创建 C# 应用程序转换项目，以便将 Oracle 数据库架构转换为 PostgreSQL 数据库架构。请务必在项目中添加包含源 Oracle 架构和目标 PostgreSQL 数据库的映射规则。有关更多信息，请参阅在 AWS SCT 中创建映射规则。

您可以在单个 AWS SCT 项目中添加多个应用程序转换项目。使用以下过程创建 C# 应用程序转换项目。

创建 C# 应用程序转换项目

1. 创建数据库转换项目，然后添加源 Oracle 数据库。有关更多信息，请参阅 创建 AWS SCT 项目 和 向 AWS SCT 项目添加数据库服务器：
2. 添加包含源 Oracle 数据库和目标 PostgreSQL 数据库的映射规则。您可以添加目标 PostgreSQL 数据库，也可以在映射规则中使用虚拟的 PostgreSQL 目标数据库平台。有关更多信息，请参阅 在 AWS SCT 中创建映射规则 和 使用虚拟目标：
3. 在视图菜单上，选择主视图。
4. 在应用程序菜单上选择新建 C# 应用程序。

   随即出现创建 C# 应用程序转换项目对话框。



5. 对于名称，输入 C# 应用程序转换项目名称。每个数据库架构转换项目都可以有一个或多个子应用程序转换项目，因此如果您添加多个项目的话，请选择一个有意义的名称。
6. 对于位置，输入您的应用程序源代码的位置。
7. 在源树状图中，选择您的应用程序所用的架构。确保此架构是映射规则的一部分。AWS SCT 以粗体突出显示作为映射规则一部分的架构。
8. 选择确定以创建 C# 应用程序转换项目。
9. 在左侧面板的应用程序节点中找到您的 C# 应用程序转换项目。

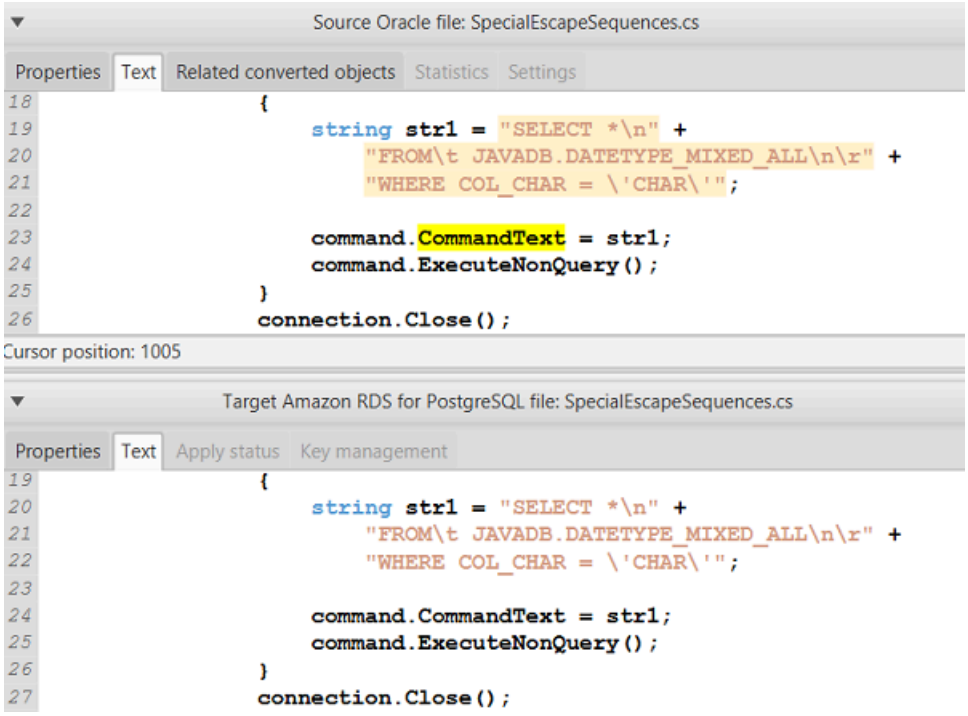
在 AWS SCT 中转换 C# 应用程序 SQL 代码

将 C# 应用程序添加到 AWS SCT 项目后，将此应用程序中的 SQL 代码转换为与目标数据库平台兼容的格式。按照以下过程在 AWS Schema Conversion Tool 中分析和转换嵌入 C# 应用程序的 SQL 代码。

转换 SQL 代码

1. 展开左侧面板中应用程序下的 C# 节点。
2. 选择要转换的应用程序，然后打开上下文（右键单击）菜单。
3. 选择转换。AWS SCT 分析源代码文件，确定应用程序逻辑，并将代码元数据加载到项目中。此代码元数据包括 C# 类、对象、方法、全局变量、接口等。

在目标数据库面板中，AWS SCT 创建与源应用程序项目相似的文件夹结构。在这里，您可以查看转换后的应用程序代码。



4. 保存转换后的应用程序代码。有关更多信息，请参阅保存转换后的应用程序代码。

您的 C# 应用程序可能包含与不同源数据库交互的 SQL 代码。您可以将其中几个源数据库迁移到 PostgreSQL。在这种情况下，请确保不要转换与已排除在迁移范围之外的数据库交互的 SQL 代码。您可以将 C# 应用程序的源文件排除在转换范围之外。为此，请清除要排除在转换范围之外的文件名的复选框。

更改转换范围后，AWS SCT 仍会分析 C# 应用程序的所有源文件的 SQL 代码。然后，AWS SCT 将您排除在转换范围之外的所有源文件复制到目标文件夹。通过此操作，可以在保存转换后的应用程序文件后生成应用程序。

使用 AWS SCT 保存转换后的应用程序代码

使用以下过程保存转换后的应用程序代码。

保存转换后的应用程序代码

1. 展开目标数据库面板中应用程序下的 C# 节点。
2. 选择转换后的应用程序，然后选择保存。

3. 输入保存转换后的应用程序代码的文件夹路径，然后选择选择文件夹。

在 AWS SCT 中管理 C# 应用程序转换项目

您可以添加多个 C# 应用程序转换项目、更新 AWS SCT 项目中的应用程序代码或从 AWS SCT 项目中删除 C# 转换项目。

添加其他 C# 应用程序转换项目

1. 展开左侧面板中的应用程序节点。
2. 选择 C# 节点，然后打开上下文 (右键单击)菜单。
3. 选择 New application（新建应用程序）。
4. 输入创建新的 C# 应用程序转换项目所需的信息。有关更多信息，请参阅创建 C# 应用程序转换项目。

更改源应用程序代码后，将其上传到 AWS SCT 项目中。

上传更新的应用程序代码

1. 展开左侧面板中应用程序下的 C# 节点。
2. 选择要更新的应用程序，然后打开上下文（右键单击）菜单。
3. 选择刷新，然后选择是。

   AWS SCT 从源文件上传您的应用程序代码并删除转换结果。要保留您在 AWS SCT 中进行的代码更改和转换结果，请创建一个新的 C# 转换项目。

删除 C# 应用程序转换项目

1. 展开左侧面板中应用程序下的 C# 节点。
2. 选择要删除的应用程序，然后打开上下文（右键单击）菜单。
3. 选择删除，然后选择确定。

在 AWS SCT 中创建 C# 应用程序转换评估报告

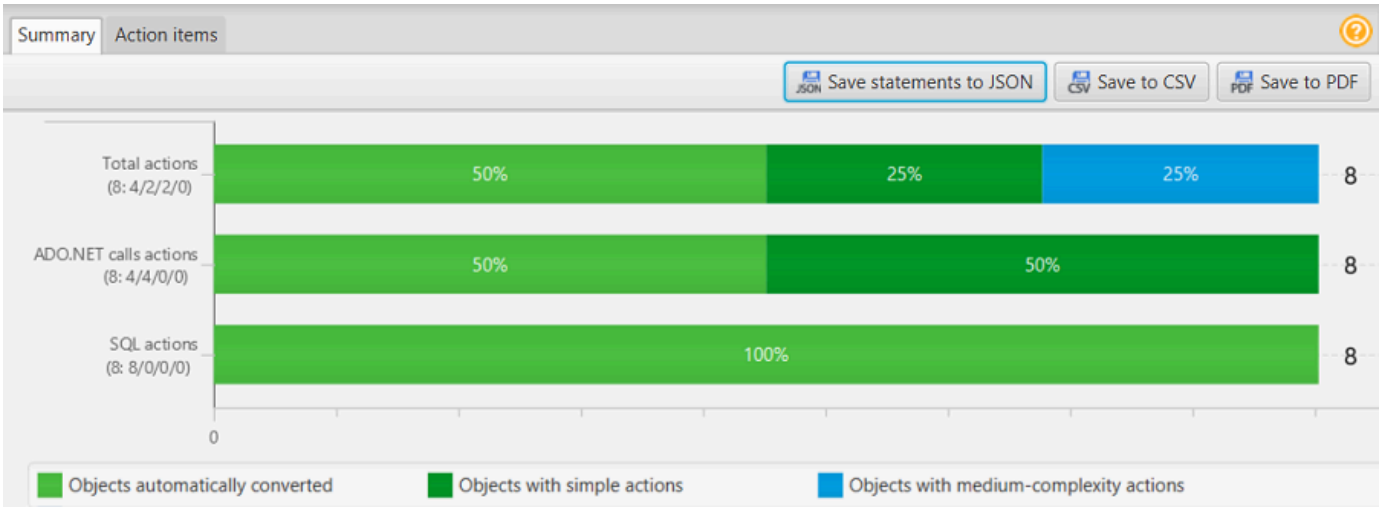
C# 应用程序转换评估报告提供了有关将 C# 应用程序中嵌入的 SQL 代码转换为与目标数据库兼容的格式的信息。评估报告提供了所有 SQL 执行点和所有源代码文件转换的详细信息。评估报告还包括针对 AWS SCT 无法转换的 SQL 代码的操作项。

使用以下过程创建 C# 应用程序转换评估报告。

创建 C# 应用程序转换评估报告

1. 展开左侧面板中应用程序下的 C# 节点。
2. 选择要转换的应用程序，然后打开上下文（右键单击）菜单。
3. 选择转换。
4. 在视图菜单上，选择评估报告视图。
5. 查看摘要选项卡。

   如下所示的摘要选项卡显示了 C# 应用程序评估报告中的执行摘要信息。它显示所有 SQL 执行点和所有源代码文件的转换结果。



6. 选择将语句保存为 JSON，将从 C# 应用程序中提取的 SQL 代码另存为 JSON 文件。
7. （可选）将报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。

   - 选择右上角的保存为 PDF，将报告另存为 PDF 文件。

     PDF 文件包含应用程序转换的执行摘要、操作项和建议。
   - 选择右上角的保存为 CSV，将报告另存为 CSV 文件。

     CSV 文件包含操作项、推荐的操作以及转换 SQL 代码所需的估计人工操作的复杂性。

使用 AWS SCT 转换 C++ 应用程序中的 SQL 代码

对于 Oracle 到 PostgreSQL 的转换，可以使用 AWS SCT 转换嵌入到 C++ 应用程序中的 SQL 代码。这个特定的 C++ 应用程序转换器了解应用程序逻辑。它收集位于不同应用程序对象中的语句，例如函数、参数、局部变量等。

由于这种深入的分析，C++ 应用程序 SQL 代码转换器比通用转换器的转换结果更好。

在 AWS SCT 中创建 C++ 应用程序转换项目

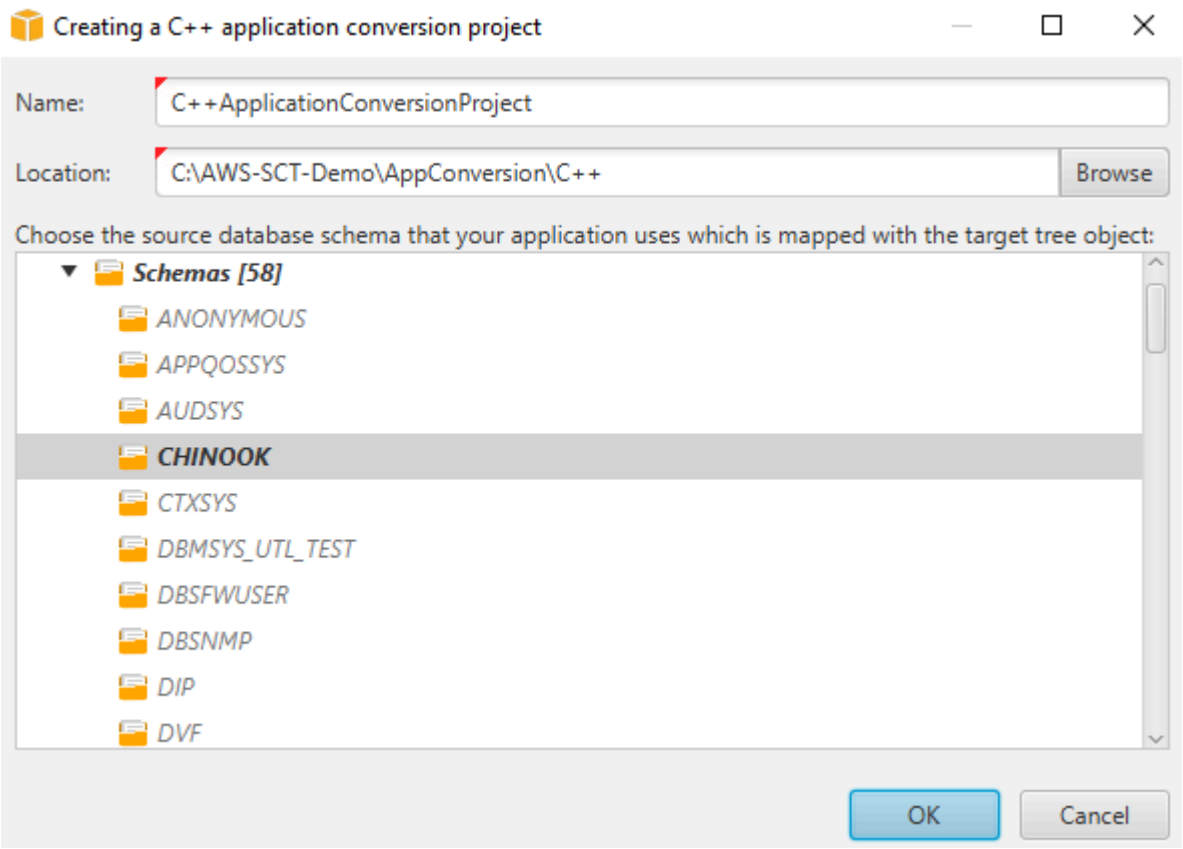
您可以仅创建 C++ 应用程序转换项目，以便将 Oracle 数据库架构转换为 PostgreSQL 数据库架构。请务必在项目中添加包含源 Oracle 架构和目标 PostgreSQL 数据库的映射规则。有关更多信息，请参阅在 AWS SCT 中创建映射规则。

您可以在单个 AWS SCT 项目中添加多个应用程序转换项目。

创建 C++ 应用程序转换项目

1. 创建数据库转换项目，然后添加源 Oracle 数据库。有关更多信息，请参阅 创建 AWS SCT 项目 和 向 AWS SCT 项目添加数据库服务器：
2. 添加包含源 Oracle 数据库和目标 PostgreSQL 数据库的映射规则。您可以添加目标 PostgreSQL 数据库，也可以在映射规则中使用虚拟的 PostgreSQL 目标数据库平台。有关更多信息，请参阅 在 AWS SCT 中创建映射规则 和 使用虚拟目标：
3. 在视图菜单上，选择主视图。
4. 在应用程序菜单上，选择新建 C++ 应用程序。

   随即出现创建 C++ 应用程序转换项目对话框。



5. 对于名称，输入 C++ 应用程序转换项目名称。每个数据库架构转换项目都可以有一个或多个子应用程序转换项目，因此如果您添加多个项目的话，请选择一个有意义的名称。
6. 对于位置，输入您的应用程序源代码的位置。
7. 在源树状图中，选择您的应用程序所用的架构。确保此架构是映射规则的一部分。AWS SCT 以粗体突出显示作为映射规则一部分的架构。
8. 选择确定以创建 C++ 应用程序转换项目。
9. 在左侧面板的应用程序节点中找到您的 C++ 应用程序转换项目。

在 AWS SCT 中转换 C++ 应用程序 SQL 代码

将 C++ 应用程序添加到 AWS SCT 项目后，将此应用程序中的 SQL 代码转换为与目标数据库平台兼容的格式。按照以下过程在 AWS SCT 中分析和转换 C++ 应用程序中嵌入的 SQL 代码。

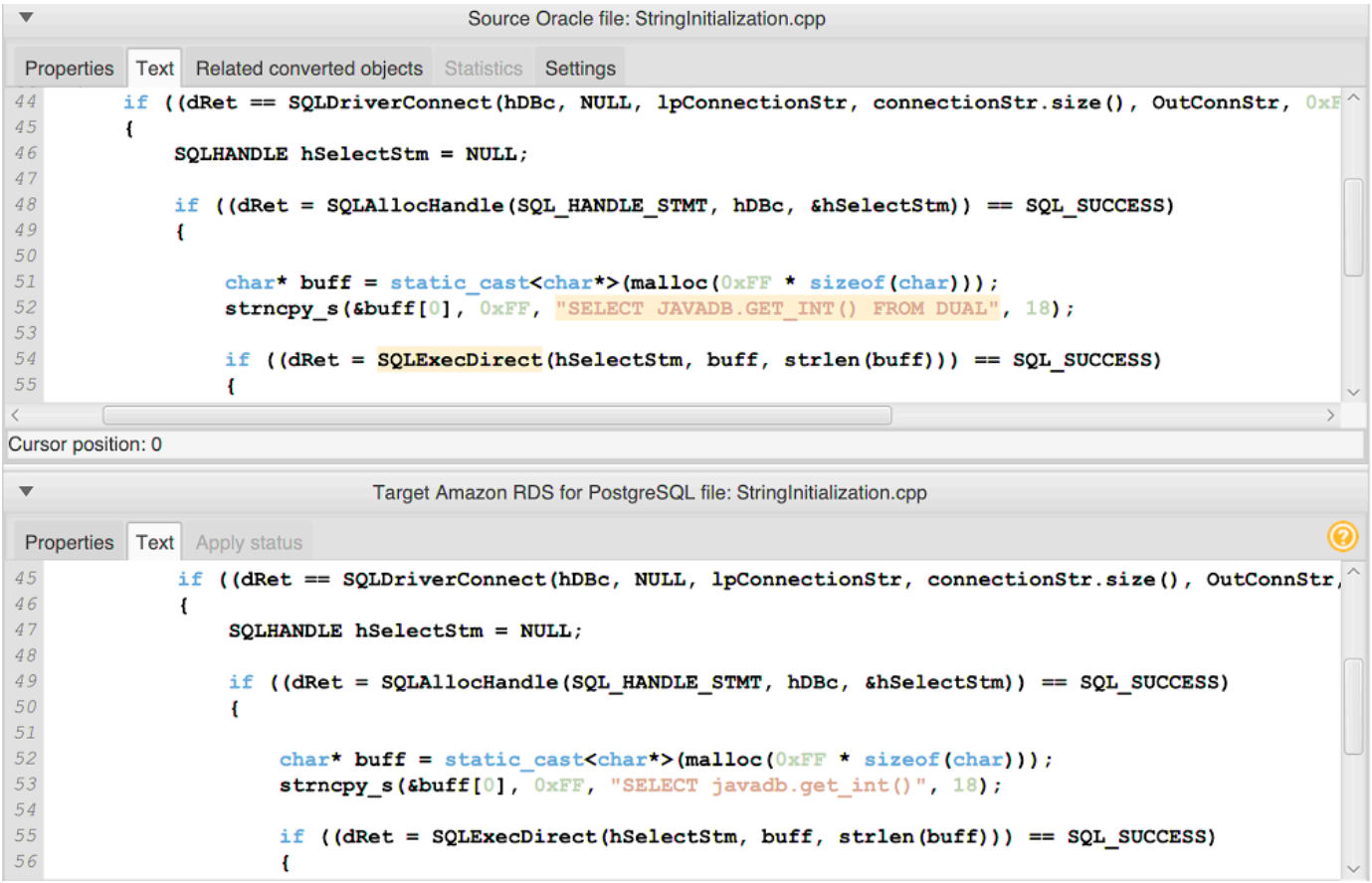
转换 SQL 代码

1. 展开左侧面板中应用程序下的 C++ 节点，然后选择要转换的应用程序。

2. 在源 Oracle 应用程序项目中，选择设置。查看和编辑所选 C++ 应用程序的转换设置。您还可以为添加到 AWS SCT 项目中的所有 C++ 应用程序指定转换设置。有关更多信息，请参阅管理 C++ 应用程序转换项目。
3. 对于编译器类型，请选择用于 C++ 应用程序源代码的编译器。AWS SCT 支持以下 C++ 编译器：Microsoft Visual C++、GCC（GNU 编译器套件）以及 Clang。默认选项是 Microsoft Visual C++。
4. 对于用户定义的宏，请输入包含来自 C++ 项目的用户定义宏的文件路径。确保该文件具有以下结构：`#define name value`。在上述示例中，`value` 是可选参数。此可选参数的默认值为 `1`。

   要创建此文件，请在 Microsoft Visual Studio 中打开项目，然后选择项目、属性、C/C++ 和预处理器。对于预处理器定义，选择编辑，然后将名称和值复制到新的文本文件中。然后，为文件中的每个字符串添加以下前缀：`#define` 。
5. 对于外部包含目录，请输入包含您在 C++ 项目中使用的外部库的文件夹的路径。
6. 在左窗格中，选择要转换的应用程序，然后打开上下文（右键单击）菜单。
7. 选择转换。AWS SCT 分析源代码文件，确定应用程序逻辑，并将代码元数据加载到项目中。此代码元数据包括 C++ 类、对象、方法、全局变量、接口等。

   在目标数据库面板中，AWS SCT 创建与源应用程序项目相似的文件夹结构。您可以在此查看转换后的应用程序代码，如下所示。



8. 保存转换后的应用程序代码。有关更多信息，请参阅保存转换后的应用程序代码。

使用 AWS SCT 保存转换后的应用程序代码

使用以下过程保存转换后的应用程序代码。

保存转换后的应用程序代码

1. 展开目标数据库面板中应用程序 下的 C++ 节点。
2. 选择转换后的应用程序，然后选择保存。
3. 输入保存转换后的应用程序代码的文件夹路径，然后选择选择文件夹。

在 AWS SCT 中管理 C++ 应用程序转换项目

您可以添加多个 C++ 应用程序转换项目、编辑转换设置、更新 C++ 应用程序代码或从 AWS SCT 项目中删除 C++ 转换项目。

添加其他 C++ 应用程序转换项目

1. 展开左侧面板中的应用程序节点。
2. 选择 C++ 节点，然后打开上下文（右键单击）菜单。
3. 选择 New application（新建应用程序）。
4. 输入创建新的 C++ 应用程序转换项目所需的信息。有关更多信息，请参阅创建 C++ 应用程序转换项目。

您可以为 AWS SCT 项目中的所有 C++ 应用程序转换项目指定转换设置。

编辑所有 C++ 应用程序的转换设置

1. 在设置菜单上，选择项目设置，然后选择 应用程序转换。
2. 对于编译器类型，请选择用于 C++ 应用程序源代码的编译器。AWS SCT 支持以下 C++ 编译器：Microsoft Visual C++、GCC（GNU 编译器套件）以及 Clang。默认选项是 Microsoft Visual C++。
3. 对于用户定义的宏，请输入包含来自 C++ 项目的用户定义宏的文件路径。确保该文件具有以下结构：`#define name value`。在上述示例中，`value` 是可选参数。此可选参数的默认值为 `1`。

   要创建此文件，请在 Microsoft Visual Studio 中打开项目，然后选择项目、属性、C/C++ 和预处理器。对于预处理器定义，选择编辑，然后将名称和值复制到新的文本文件中。然后，为文件中的每个字符串添加以下前缀：`#define` 。
4. 对于外部包含目录，请输入包含您在 C++ 项目中使用的外部库的文件夹的路径。
5. 选择确定以保存项目设置并关闭窗口。

或者，您可以为每个 C++ 应用程序转换项目指定转换设置。有关更多信息，请参阅转换 C++ 应用程序 SQL 代码。

更改源应用程序代码后，将其上传到 AWS SCT 项目中。

上传更新的应用程序代码

1. 展开左侧面板中应用程序下的 C++ 节点。
2. 选择要更新的应用程序，然后打开上下文（右键单击）菜单。
3. 选择刷新，然后选择是。

AWS SCT 从源文件上传您的应用程序代码并删除转换结果。要保留您在 AWS SCT 中进行的代码更改和转换结果，请创建一个新的 C++ 转换项目。

此外，AWS SCT 还会删除您为所选应用程序指定的应用程序转换设置。上传更新的应用程序代码后，AWS SCT 应用项目设置中的默认值。

删除 C++ 应用程序转换项目

1. 展开左侧面板中应用程序下的 C++ 节点。
2. 选择要删除的应用程序，然后打开上下文（右键单击）菜单。
3. 选择删除，然后选择确定。

在 AWS SCT 中创建 C++ 应用程序转换评估报告

C++ 应用程序转换评估报告提供了有关将 C++ 应用程序中嵌入的 SQL 代码转换为与目标数据库兼容的格式的信息。评估报告提供了所有 SQL 执行点和所有源代码文件转换的详细信息。评估报告还包括针对 AWS SCT 无法转换的 SQL 代码的操作项。

创建 C++ 应用程序转换评估报告

1. 展开左侧面板中应用程序下的 C++ 节点。
2. 选择要转换的应用程序，然后打开上下文（右键单击）菜单。
3. 选择转换。
4. 在视图菜单上，选择评估报告视图。
5. 查看摘要选项卡。

   摘要选项卡显示了 C++ 应用程序评估报告中的执行摘要信息。它显示所有 SQL 执行点和所有源代码文件的转换结果。
6. 选择将语句保存为 JSON，将从 Java 应用程序中提取的 SQL 代码另存为 JSON 文件。
7. （可选）将报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。

   - 选择右上角的保存为 PDF，将报告另存为 PDF 文件。

     PDF 文件包含应用程序转换的执行摘要、操作项和建议。
   - 选择右上角的保存为 CSV，将报告另存为 CSV 文件。

CSV 文件包含操作项、推荐的操作以及转换 SQL 代码所需的估计人工操作的复杂性。

使用 AWS SCT 转换 Java 应用程序中的 SQL 代码

对于 Oracle 到 PostgreSQL 的转换，可以使用 AWS Schema Conversion Tool 转换嵌入到 Java 应用程序中的 SQL 代码。这个特定的 Java 应用程序转换器了解应用程序逻辑。它收集位于不同应用程序对象中的语句，例如函数、参数、局部变量等。

由于这种深入的分析，与通用转换器相比，Java 应用程序 SQL 代码转换器的转换结果更好。

如果您的 Java 应用程序使用 MyBatis 框架与数据库进行交互，则您可以使用 AWS SCT 转换嵌入到 MyBatis XML 文件和注释中的 SQL 语句。要了解这些 SQL 语句的逻辑，AWS SCT 使用 MyBatis 配置文件。AWS SCT 可以在您的应用程序文件夹中自动发现此文件，也可以手动输入该文件的路径。

在 AWS SCT 中创建 Java 应用程序转换项目

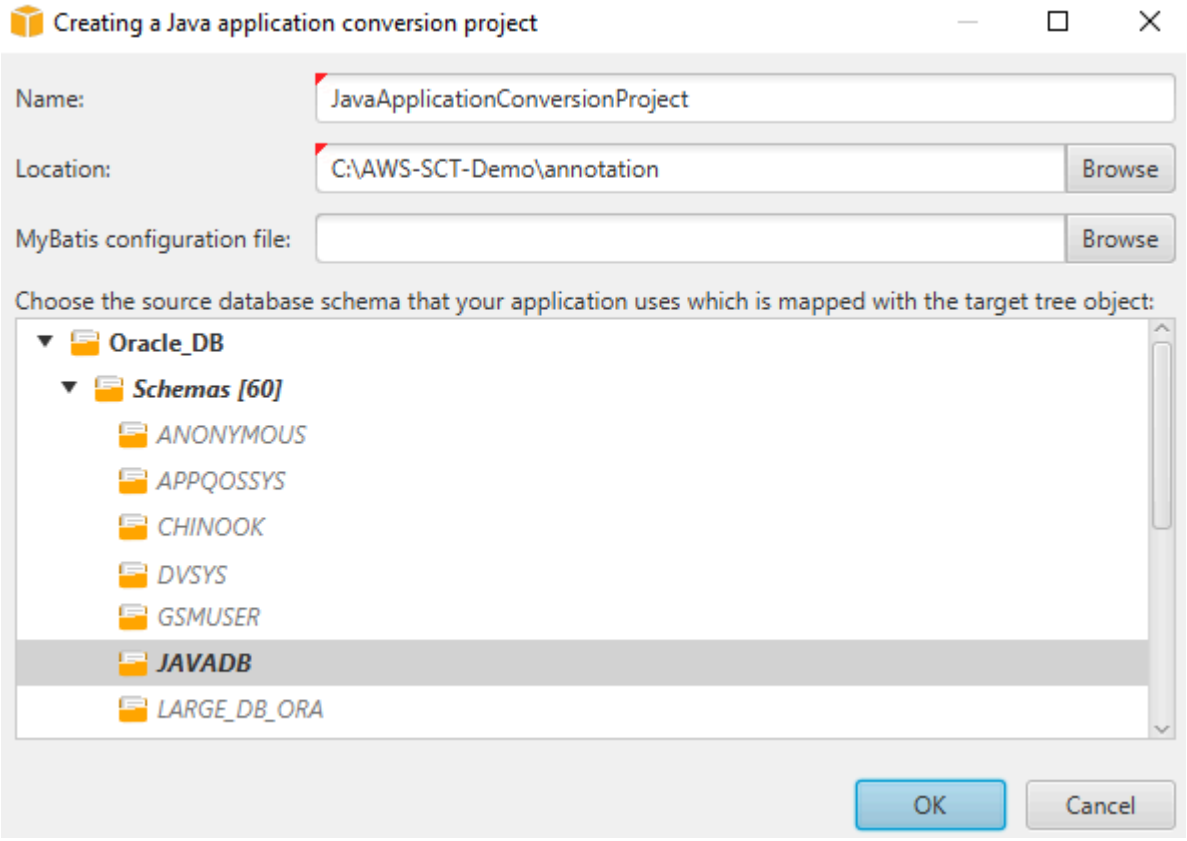
您可以仅创建 Java 应用程序转换项目，以便将 Oracle 数据库架构转换为 PostgreSQL 数据库架构。请务必在项目中添加包含源 Oracle 架构和目标 PostgreSQL 数据库的映射规则。有关更多信息，请参阅在 AWS SCT 中创建映射规则。

您可以在单个 AWS SCT 项目中添加多个应用程序转换项目。使用以下过程创建 Java 应用程序转换项目。

创建 Java 应用程序转换项目

1. 创建数据库转换项目，然后添加源 Oracle 数据库。有关更多信息，请参阅 创建 AWS SCT 项目 和 向 AWS SCT 项目添加数据库服务器：
2. 添加包含源 Oracle 数据库和目标 PostgreSQL 数据库的映射规则。您可以添加目标 PostgreSQL 数据库，也可以在映射规则中使用虚拟的 PostgreSQL 目标数据库平台。有关更多信息，请参阅 在 AWS SCT 中创建映射规则 和 使用虚拟目标：
3. 在视图菜单上，选择主视图。
4. 在应用程序菜单上，选择新建 Java 应用程序。

   随即出现创建 Java 应用程序转换项目对话框。



5. 对于名称，输入 Java 应用程序转换项目名称。每个数据库架构转换项目都可以有一个或多个子应用程序转换项目，因此如果您添加多个项目的话，请选择一个有意义的名称。
6. 对于位置，输入您的应用程序源代码的位置。
7. （可选）对于 MyBatis 配置文件，请输入 MyBatis 配置文件的路径。AWS SCT 会扫描您的应用程序文件夹以自动发现此文件。如果此文件不在您的应用程序文件夹中，或者您使用了多个配置文件，请手动输入路径。
8. 在源树状图中，选择应用程序所用的架构。确保此架构是映射规则的一部分。AWS SCT 以粗体突出显示作为映射规则一部分的架构。
9. 选择确定以创建 Java 应用程序转换项目。
10. 在左侧面板的应用程序节点中找到您的 Java 应用程序转换项目。

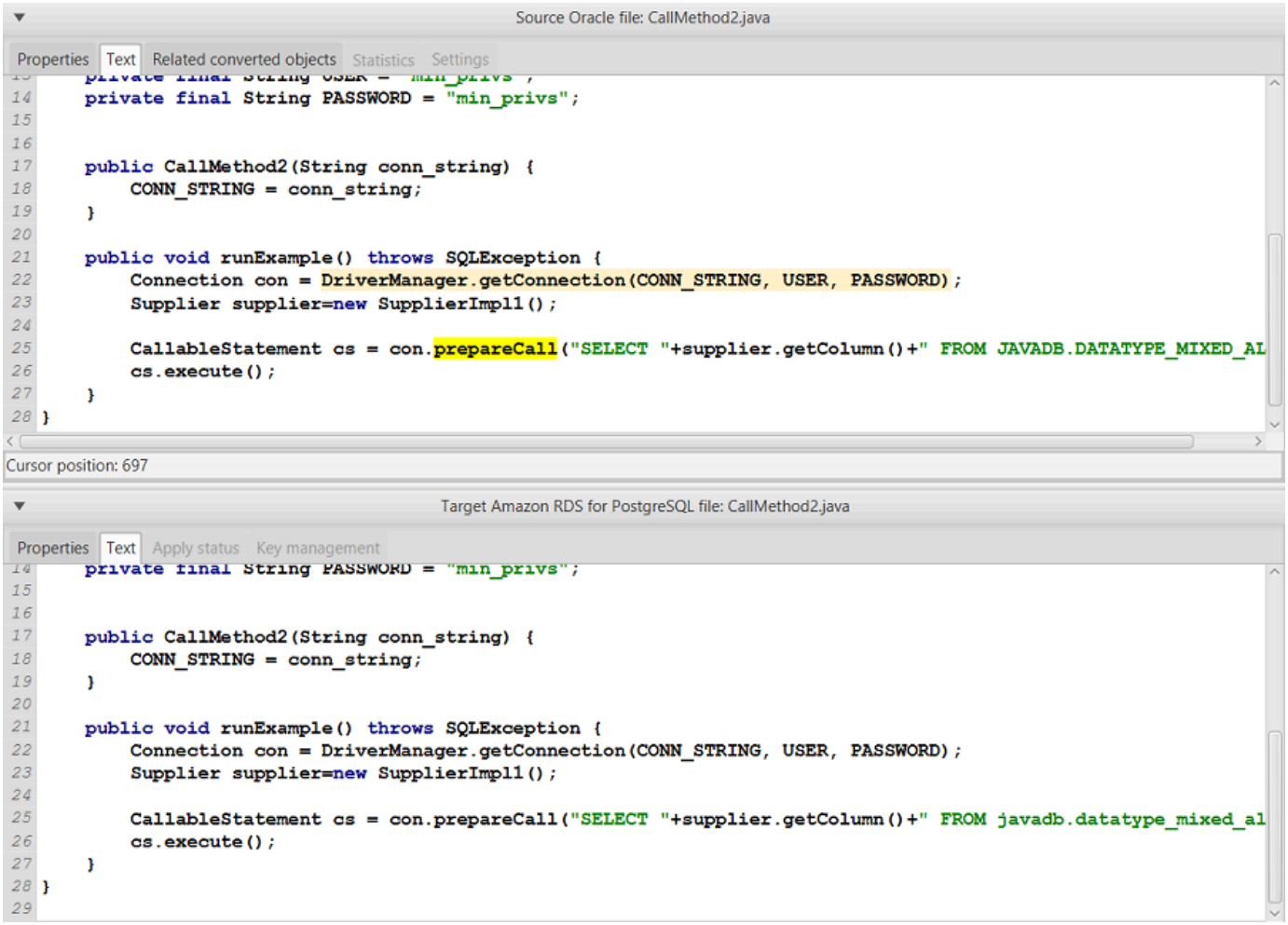
在 AWS SCT 中转换 Java 应用程序 SQL 代码

将 Java 应用程序添加到 AWS SCT 项目后，将此应用程序中的 SQL 代码转换为与目标数据库平台兼容的格式。按照以下过程在 AWS Schema Conversion Tool 中分析和转换嵌入 Java 应用程序的 SQL 代码。

转换 SQL 代码

1. 展开左侧面板中应用程序下的 Java 节点。
2. 选择要转换的应用程序，然后打开上下文（右键单击）菜单。
3. 选择转换。AWS SCT 分析源代码文件，确定应用程序逻辑，并将代码元数据加载到项目中。此代码元数据包括 Java 类、对象、方法、全局变量、接口等。

   在目标数据库面板中，AWS SCT 创建与源应用程序项目相似的文件夹结构。在这里，您可以查看转换后的应用程序代码。



4. 保存转换后的应用程序代码。有关更多信息，请参阅保存转换后的应用程序代码。

您的 Java 应用程序可能包含与不同源数据库交互的 SQL 代码。您可以将其中几个源数据库迁移到 PostgreSQL。在这种情况下，请确保不要转换与已排除在迁移范围之外的数据库交互的 SQL 代码。您可以将 Java 应用程序的源文件排除在转换范围之外。为此，请清除要排除在转换范围之外的文件名的复选框。

更改转换范围后，AWS SCT 仍会分析 Java 应用程序的所有源文件的 SQL 代码。然后，AWS SCT 将您排除在转换范围之外的所有源文件复制到目标文件夹。通过此操作，可以在保存转换后的应用程序文件后生成应用程序。

使用 AWS SCT 保存转换后的应用程序代码

使用以下过程保存转换后的应用程序代码。

保存转换后的应用程序代码

1. 展开目标数据库面板中应用程序下的 Java 节点。
2. 选择转换后的应用程序，然后选择保存。
3. 输入保存转换后的应用程序代码的文件夹路径，然后选择选择文件夹。

如果您的源 Java 应用程序使用 MyBatis 框架，请务必更新配置文件以使用新数据库。

在 AWS SCT 中管理 Java 应用程序转换项目

您可以添加多个 Java 应用程序转换项目、更新 AWS SCT 项目中的应用程序代码或从 AWS SCT 项目中删除 Java 转换项目。

添加其他 Java 应用程序转换项目

1. 展开左侧面板中的应用程序节点。
2. 选择 Java 节点，并打开上下文 (右键单击) 菜单。
3. 选择 New application（新建应用程序）。
4. 输入创建新的 Java 应用程序转换项目所需的信息。有关更多信息，请参阅创建 Java 应用程序转换项目。

更改源应用程序代码后，将其上传到 AWS SCT 项目中。

上传更新的应用程序代码

1. 展开左侧面板中应用程序下的 Java 节点。
2. 选择要更新的应用程序，然后打开上下文（右键单击）菜单。
3. 选择刷新，然后选择是。

AWS SCT 从源文件上传您的应用程序代码并删除转换结果。要保留您在 AWS SCT 中进行的代码更改和转换结果，请创建一个新的 Java 转换项目。

如果您的源 Java 应用程序使用 MyBatis 框架，则 AWS SCT 使用 MyBatis 配置文件来解析您的 SQL 代码。更改此文件后，将其上传到 AWS SCT 项目中。

编辑 MyBatis 配置文件的路径

1. 展开左侧面板中应用程序下的 Java 节点。
2. 选择您的应用程序，然后选择设置。
3. 选择浏览，然后选择 MyBatis 配置文件。
4. 选择 Apply（应用）。
5. 在左侧面板中，选择应用程序，打开上下文（右键单击）菜单，然后选择刷新。

删除 Java 应用程序转换项目

1. 展开左侧面板中应用程序下的 Java 节点。
2. 选择要删除的应用程序，然后打开上下文（右键单击）菜单。
3. 选择删除，然后选择确定。

在 AWS SCT 中创建 Java 应用程序转换评估报告

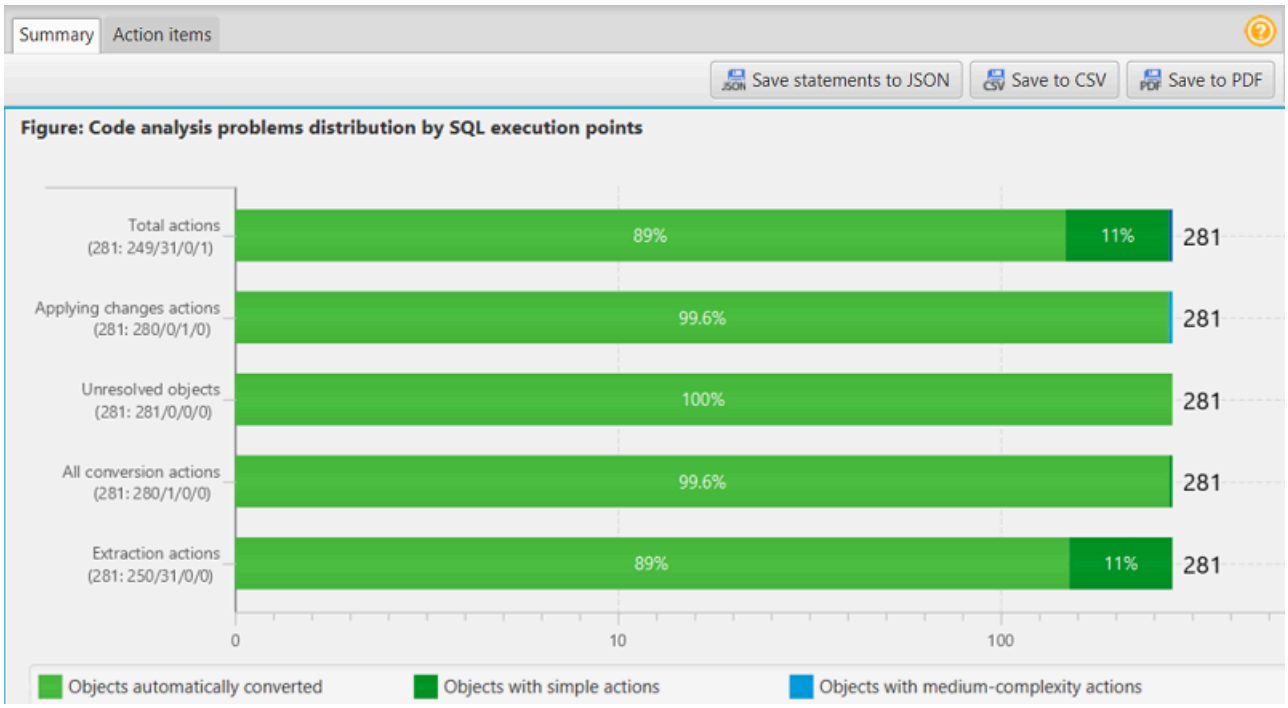
Java 应用程序转换评估报告提供了有关将 Java 应用程序中嵌入的 SQL 代码转换为与目标数据库兼容的格式的信息。评估报告提供了所有 SQL 执行点和所有源代码文件转换的详细信息。评估报告还包括针对 AWS SCT 无法转换的 SQL 代码的操作项。

使用以下过程创建 Java 应用程序转换评估报告。

创建 Java 应用程序转换评估报告

1. 展开左侧面板中应用程序下的 Java 节点。
2. 选择要转换的应用程序，然后打开上下文（右键单击）菜单。
3. 选择转换。
4. 在视图菜单上，选择评估报告视图。
5. 查看 Summary 选项卡。

如下所示的摘要选项卡显示了 Java 应用程序评估报告中的执行摘要信息。它显示所有 SQL 执行点和所有源代码文件的转换结果。



6. 选择将语句保存为 JSON，将从 Java 应用程序中提取的 SQL 代码另存为 JSON 文件。
7. （可选）将报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。

   - 选择右上角的保存为 PDF，将报告另存为 PDF 文件。

     PDF 文件包含应用程序转换的执行摘要、操作项和建议。
   - 选择右上角的保存为 CSV，将报告另存为 CSV 文件。

     CSV 文件包含操作项、推荐的操作以及转换 SQL 代码所需的估计人工操作的复杂性。

使用 AWS SCT 转换 Pro*C 应用程序中的 SQL 代码

对于 Oracle 到 PostgreSQL 的转换，可以使用 AWS Schema Conversion Tool（AWS SCT）转换嵌入到 Pro*C 应用程序中的 SQL 代码。这个特定的 Pro*C 应用程序转换器了解应用程序逻辑。它收集位于不同应用程序对象中的语句，例如函数、参数、局部变量等。

由于这种深入的分析，与通用转换器相比，Pro*C 应用程序 SQL 代码转换器的转换结果更好。

在 AWS SCT 中创建 Pro*C 应用程序转换项目

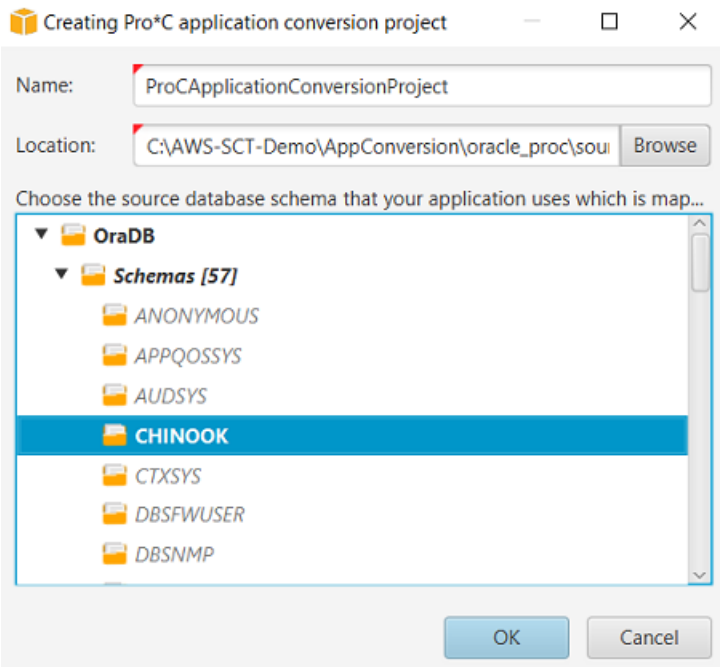
您可以仅创建 Pro*C 应用程序转换项目，以便将 Oracle 数据库架构转换为 PostgreSQL 数据库架构。请务必在项目中添加包含源 Oracle 架构和目标 PostgreSQL 数据库的映射规则。有关更多信息，请参阅在 AWS SCT 中创建映射规则。

您可以在单个 AWS SCT 项目中添加多个应用程序转换项目。使用以下过程创建 Pro*C 应用程序转换项目。

创建 Pro*C 应用程序转换项目

1. 创建数据库转换项目，然后添加源 Oracle 数据库。有关更多信息，请参阅 创建 AWS SCT 项目 和 向 AWS SCT 项目添加数据库服务器：
2. 添加包含源 Oracle 数据库和目标 PostgreSQL 数据库的映射规则。您可以添加目标 PostgreSQL 数据库，也可以在映射规则中使用虚拟的 PostgreSQL 目标数据库平台。有关更多信息，请参阅 在 AWS SCT 中创建映射规则 和 使用虚拟目标：
3. 在视图菜单上，选择主视图。
4. 在应用程序菜单上，选择新建 Pro*C 应用程序。

   随即出现创建 Pro*C 应用程序转换项目对话框。



5. 对于名称，输入 Pro*C 应用程序转换项目名称。每个数据库架构转换项目都可以有一个或多个子应用程序转换项目，因此如果您添加多个项目的话，请选择一个有意义的名称。
6. 对于位置，输入您的应用程序源代码的位置。

7. 在源树状图中，选择应用程序所用的架构。确保此架构是映射规则的一部分。AWS SCT 以粗体突出显示作为映射规则一部分的架构。
8. 选择确认以创建 Pro*C 应用程序转换项目。
9. 在左侧面板的应用程序 节点中找到您的 Pro*C 应用程序转换项目。

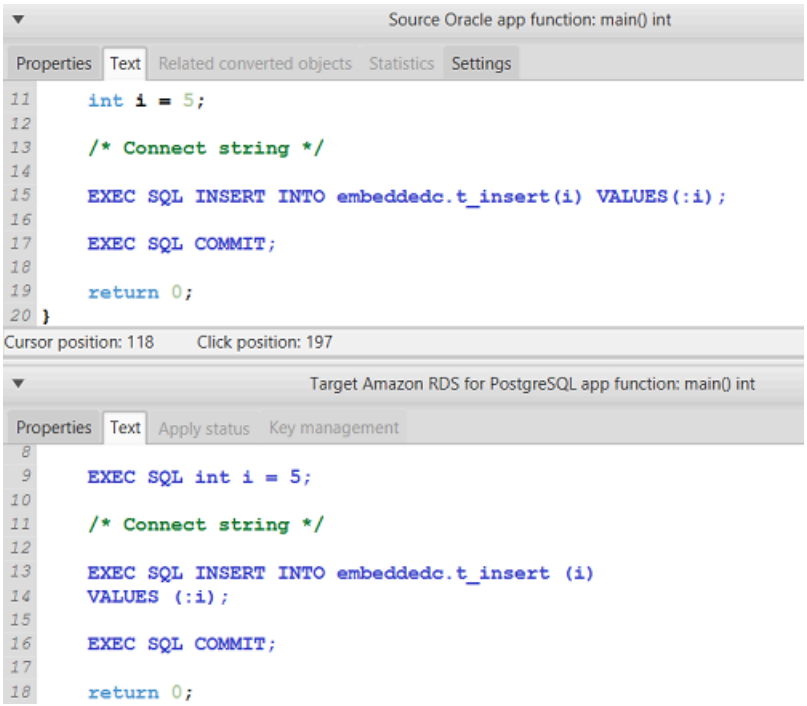
在 AWS SCT 中转换 Pro*C 应用程序 SQL 代码

将 Pro*C 应用程序添加到 AWS SCT 项目后，将该应用程序中的 SQL 代码转换为与目标数据库平台兼容的格式。按照以下过程在 AWS Schema Conversion Tool 中分析和转换中嵌入 Pro*C 应用程序中的 SQL 代码。

转换 SQL 代码

1. 展开左侧面板中应用程序下的 Pro*C 节点。
2. 选择要转换的应用程序，然后选择设置。
   a. 在全局标头文件路径中，输入应用程序项目使用的标头文件路径。
   b. 选择将所有未解析的主机变量解释为，以查看转换后的代码中所有未解析的变量。
   c. 选择使用扩展包中的固定宽度的字符串转换函数，以便在转换后的 SQL 代码中使用扩展包函数。 AWS SCT 在应用程序项目中包含扩展包文件。
   d. 选择将匿名 PL/SQL 块转换为独立的 SQL 调用或存储函数，在目标数据库中为所有匿名 PL/SQL 块创建存储过程。然后，AWS SCT 将这些存储过程的运行包括在转换后的应用程序代码中。
   e. 选择使用自定义游标流可改进 Oracle 数据库游标的转换。
3. 在左侧面板中，选择要转换的应用程序，然后打开上下文（右键单击）菜单。
4. 选择转换。AWS SCT 分析源代码文件，确定应用程序逻辑，并将代码元数据加载到项目中。此代码元数据包括 Pro*C 类、对象、方法、全局变量、接口等。

   在目标数据库面板中，AWS SCT 创建与源应用程序项目相似的文件夹结构。在这里，您可以查看转换后的应用程序代码。



5. 保存转换后的应用程序代码。有关更多信息，请参阅编辑和保存转换后的 SQL 代码。

使用 AWS SCT 编辑和保存转换后的 SQL 代码

您可以编辑转换后的 SQL 语句，并使用 AWS SCT 将编辑后的代码嵌入到转换后的 Pro*C 应用程序代码中。您可使用以下过程编辑转换后的 SQL 代码。

编辑转换后的 SQL 代码

1. 展开左侧面板中应用程序下的 Pro*C 节点。
2. 选择要转换的应用程序，打开上下文（右键单击）菜单，然后选择转换。
3. 在视图菜单上，选择评估报告视图。
4. 选择将语句保存为 CSV，将从 Pro*C 应用程序中提取的 SQL 代码保存为 CSV 文件。
5. 输入 CSV 文件的名称以保存提取的 SQL 代码，然后选择保存。
6. 编辑提取的 SQL 代码。
7. 在视图菜单上，选择主视图。
8. 展开目标数据库面板中应用程序下的 Pro*C 节点。
9. 选择转换后的应用程序，打开上下文（右键单击）菜单，然后选择从 CSV 导入语句。
10. 选择是，然后选择包含已编辑的 SQL 代码的文件，然后选择打开。

AWS SCT 将转换后的 SQL 语句分成多个部分，然后将其放入源应用程序代码的相应对象中。使用以下过程保存转换后的应用程序代码。

保存转换后的应用程序代码

1. 展开目标数据库面板中应用程序下的 Pro*C 节点。
2. 选择转换后的应用程序，然后选择保存。
3. 输入保存转换后的应用程序代码的文件夹路径，然后选择选择文件夹。

在 AWS SCT 中管理 Pro*C 应用程序转换项目

您可以添加多个 Pro*C 应用程序转换项目、更新 AWS SCT 项目中的应用程序代码或从 AWS SCT 项目中移除 Pro*C 转换项目。

添加其他 Pro*C 应用程序转换项目

1. 展开左侧面板中的应用程序节点。
2. 选择 Pro*C 节点，然后打开上下文（右键单击）菜单。
3. 选择 New application（新建应用程序）。
4. 输入创建新的 Pro*C 应用程序转换项目所需的信息。有关更多信息，请参阅创建 Pro*C 应用程序转换项目。

更改源应用程序代码后，将其上传到 AWS SCT 项目中。

上传更新的应用程序代码

1. 展开左侧面板中应用程序下的 Pro*C 节点。
2. 选择要更新的应用程序，然后打开上下文（右键单击）菜单。
3. 选择刷新，然后选择是。

   AWS SCT 从源文件上传您的应用程序代码并删除转换结果。要保留您在 AWS SCT 中进行的代码更改和转换结果，请创建一个新的 Pro*C 转换项目。

移除 Pro*C 应用程序转换项目

1. 展开左侧面板中应用程序下的 Pro*C 节点。
2. 选择要删除的应用程序，然后打开上下文（右键单击）菜单。

3. 选择删除，然后选择确定。

在 AWS SCT 中创建 Pro*C 应用程序转换评估报告

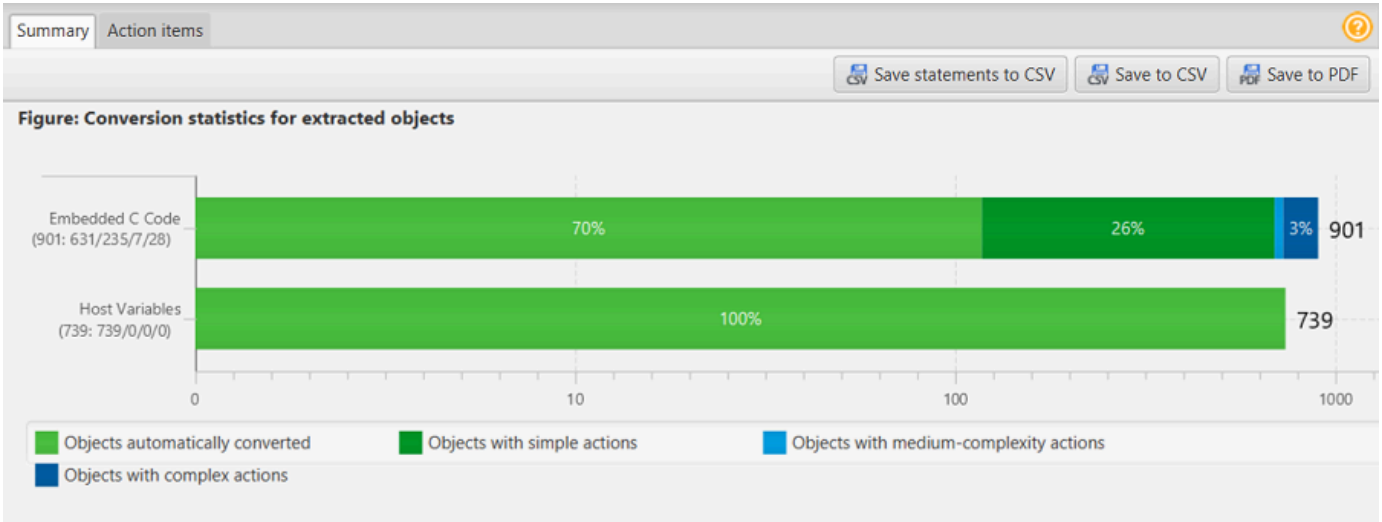
Pro*C 应用程序转换评估报告提供了有关将 Pro*C 应用程序中嵌入的 SQL 代码转换为与目标数据库兼容的格式的信息。评估报告提供了所有 SQL 执行点和所有源代码文件转换的详细信息。评估报告还包括针对 AWS SCT 无法转换的 SQL 代码的操作项。

使用以下过程创建 Pro*C 应用程序转换评估报告。

创建 Pro*C 应用程序转换评估报告

1. 展开左侧面板中应用程序下的 Pro*C 节点。
2. 选择要转换的应用程序，然后打开上下文（右键单击）菜单。
3. 选择转换。
4. 在视图菜单上，选择评估报告视图。
5. 查看 Summary 选项卡。

   如下所示的摘要选项卡显示了 Pro*C 应用程序评估报告中的执行摘要信息。它显示所有 SQL 执行点和所有源代码文件的转换结果。



6. 选择将语句保存到 CSV，将从 Pro*C 应用程序中提取的 SQL 代码保存为逗号分隔值（CSV）文件。
7. （可选）将报告的本地副本另存为 PDF 文件或逗号分隔值（CSV）文件。

   - 选择右上角的保存为 PDF，将报告另存为 PDF 文件。

PDF 文件包含应用程序转换的执行摘要、操作项和建议。

- 选择右上角的保存为 CSV，将报告另存为 CSV 文件。

  CSV 文件包含操作项、推荐的操作以及转换 SQL 代码所需的估计人工操作的复杂性。

使用 AWS SCT 扩展包

AWS SCT 扩展包是一个附加模块，用于模拟源数据库中存在的函数，这些函数在将对象转换为目标数据库时是必需的。在安装 AWS SCT 扩展包之前，需要先转换数据库架构。

每个 AWS SCT 扩展包都包含以下组件：

- 数据库架构：包括用于模拟某些联机事务处理 (OLTP) 和联机分析处理 (OLAP) 数据库对象（例如序列）的 SQL 函数、过程和表。此外，还可以在源数据库 built-in-functions 中进行不支持的模拟。此架构名称采用以下格式：`aws_`*`database_engine_name`*`_ext`。
- AWS Lambda 函数（适用于某些 OLTP 数据库）-包括模拟复杂数据库功能的 AWS Lambda 函数，例如作业调度和发送电子邮件。
- OLAP 数据库的自定义库 — 包括一组 Java 和 Python 库，您可以使用这些库将微软 SQL Server 集成服务 (SSIS) 提取、转换和加载 (ETL) 脚本迁移到或。 AWS Glue AWS Glue Studio

  Java 库包括以下模块：

  - `spark-excel_2.11-0.13.1.jar`：模拟 Excel 源组件和目标组件的功能。
  - `spark-xml_2.11-0.9.0.jar`、`poi-ooxml-schemas-4.1.2.jar` 和 `xmlbeans-3.1.0.jar`：模拟 XML 源组件的功能。

  Python 库包括以下模块。

  - `sct_utils.py`：模拟源数据类型并准备 Spark SQL 查询参数。
  - `ssis_datetime.py`：模拟日期和时间内置函数。
  - `ssis_null.py`：模拟 ISNULL 和 REPLACENULL 内置函数。
  - `ssis_string.py`：模拟字符串内置函数。

  有关这些库的更多信息，请参阅为 AWS SCT 扩展包使用自定义库。

您可以通过两种方式应用 AWS SCT 扩展包：

- AWS SCT 在应用目标数据库脚本时，可以通过从上下文菜单中选择 “应用于数据库” 来自动应用扩展包。 AWS SCT 先应用扩展包，然后再应用所有其他架构对象。
- 要手动应用扩展包，请选择目标数据库，然后从上下文菜单中选择应用扩展包。对于大多数情况，自动应用已足够。但是，如果意外删除了扩展包，则可能需要手动应用该包。

每次将 AWS SCT 扩展包应用于目标数据存储时，组件都会被覆盖，并 AWS SCT 显示有关此的通知。要关闭这些通知，请选择设置、全局设置、通知，然后选择隐藏扩展包更换提醒。

要从 Microsoft SQL Server 转换到 PostgreSQL，您可以使用 AWS SCT中的 SQL Server 到 PostgreSQL 扩展包。此扩展包模拟 SQL Server 代理和 SQL Server 数据库邮件。有关更多信息，请参阅 使用扩展包在 PostgreSQL 中模拟 SQL Server Agent 和 使用扩展包在 PostgreSQL 中模拟 SQL Server 数据库邮件。

接下来，您可以找到有关使用 AWS SCT 扩展包的更多信息。

主题

使用 AWS SCT 扩展包的权限

Amazon Aurora 的 AWS SCT 扩展包使用 AWS Lambda 函数模拟邮件发送、作业调度、排队和其他操作。将 AWS SCT 扩展包应用于目标 Aurora 数据库时， AWS SCT 会创建一个新 AWS Identity and Access Management (IAM) 角色和内联 IAM 策略。接下来， AWS SCT 创建一个新的 Lambda 函数，并配置您的 Aurora 数据库集群以实现出站连接。 AWS Lambda要运行这些操作，请确保向 IAM 用户授予以下必需权限：

- `iam:CreateRole`— 为您的 AWS 账户创建新的 IAM 角色。
- `iam:CreatePolicy`— 为您的 AWS 账户创建新的 IAM 策略。
- `iam:AttachRolePolicy`：将指定策略附加到 IAM 角色。
- `iam:PutRolePolicy`：更新嵌入在 IAM 角色中的内联策略文档。
- `iam:PassRole`：将指定的 IAM 角色传递给规则引擎。
- `iam:TagRole`：将标签添加到 IAM 角色。
- `iam:TagPolicy`：将标签添加到 IAM 策略中。
- `lambda:ListFunctions`：查看您的 Lambda 函数列表。
- `lambda:ListTags`：查看您的 Lambda 函数的标签列表。

- `lambda:CreateFunction`：新建 Lambda 函数。
- `rds:AddRoleToDBCluster`：将 IAM 角色与您的 Aurora 数据库集群关联。

Amazon Redshift 的 AWS SCT 扩展包模拟了将转换后的对象应用到 Amazon Redshift 时所需的源数据仓库基础函数。在将转换后的代码应用于 Amazon Redshift 之前，必须应用 Amazon Redshift 的扩展包。为此，请将 `iam:SimulatePrincipalPolicy` 操作包含在 IAM 策略中。

AWS SCT 使用 IAM 策略模拟器检查安装 Amazon Redshift 扩展包所需的权限。即使您已正确配置 IAM 用户，IAM Policy Simulator 也会显示错误消息。这是 IAM Policy Simulator 的已知问题。此外，当 IAM 策略中没有 `iam:SimulatePrincipalPolicy` 操作时，IAM Policy Simulator 会显示一条错误消息。在这些情况下，您可以忽略错误消息，使用扩展包向导应用扩展包。有关更多信息，请参阅应用扩展包。

使用扩展包架构

转换数据库或数据仓库架构时，AWS SCT 会向您的目标数据库添加一个额外的架构。该架构用于实现将转换后的架构写入到目标数据库时必需的源数据库的 SQL 系统功能。这个额外的架构称为扩展包架构。

OLTP 数据库的扩展包架构按照源数据库命名，如下所示：

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

OLAP 数据仓库应用程序的扩展包架构按照源数据存储命名，如下所示：

- 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`

为 AWS SCT 扩展包使用自定义库

在某些情况下，AWS SCT 无法将源数据库功能转换为目标数据库中的等效功能。相关的 AWS SCT 扩展包包含自定义库，这些库可以模拟目标数据库中的某些源数据库功能。

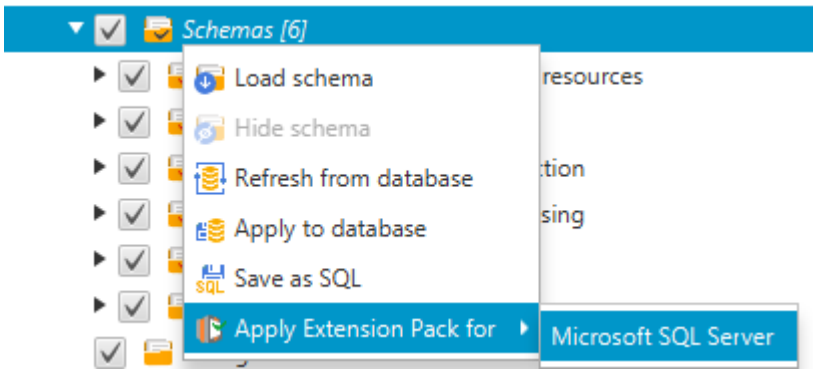
如果您要转换事务型数据库，请参阅 使用 AWS SCT 扩展包中的 AWS Lambda 函数 。

应用扩展包

您可以使用 AWS SCT 扩展包向导或在将转换后的代码应用于目标数据库时应用扩展包。

使用扩展包向导应用扩展包

1. 在目标数据库树中 AWS Schema Conversion Tool，打开上下文（右键单击）菜单，选择 A pply extension pack for，然后选择您的源数据库平台。



   扩展包向导随即出现。
2. 查看 Welcome 页面，然后选择 Next。
3. 在 AWS 配置文件页面上，执行以下操作。

   - 如果您只要重新安装扩展包架构，则选择 Skip this step for now，然后选择 Next。暂时跳过此步骤选项仅适用于联机事务处理（OLTP）数据库。
   - 如果您要上传新库，则提供凭证以连接到您的 AWS 账户。仅在转换 OLAP 数据库或 ETL 脚本时使用此步骤。如果您已 AWS CLI 安装您的 AWS Command Line Interface (AWS CLI) 凭证，则可以使用该凭证。您也可以使用之前存储在全局应用程序设置的配置文件中且与项目关联的凭证。如有必要，请选择 “导航到全局设置” 以配置不同的配置文件或将其与您的 AWS SCT 项目关联。有关更多信息，请参阅 将 AWS 服务配置文件存储在 AWS SCT 中。
4. 如果您要上传新库，请在库上传页面上选择我需要上传库。仅在转换 OLAP 数据库或 ETL 脚本时使用此步骤。接下来，提供 Amazon S3 路径，然后选择将库上传到 S3。

如果您已经上传了库，请在库上传页面上选择我已经上传了库，使用我现有 S3 存储桶。接下来，提供 Amazon S3 路径。

完成后，选择 Next 。

5. 在函数模拟页面上，选择创建扩展包。此时显示包含扩展包操作状态的消息。

   完成后，选择 Finish。

在应用转换后的代码时应用扩展包

1. 在您的 AWS 服务配置文件中指定 Amazon S3 存储桶。仅在转换 OLAP 数据库或 ETL 脚本时使用此步骤。有关更多信息，请参阅 将 AWS 服务配置文件存储在 AWS SCT 中。

   确保 Amazon S3 存储桶策略包含以下权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["s3:PutObject"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["iam:SimulatePrincipalPolicy"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["iam:GetUser"],
        "Resource": ["arn:aws:iam::111122223333:user/DataExtractionAgentName"]
      }
    ]
}
```

在前面的示例中，将 *111122223333: user/ ##### IAM ##### DataExtractionAgentName*。

2. 转换源数据仓库架构。有关更多信息，请参阅 将数据仓库架构转换为 Amazon Redshift。
3. 在右窗格中，选择转换后的架构。
4. 打开架构元素的上下文 (右键单击) 菜单，然后选择 Apply to database。
5. AWS SCT 生成包含所需组件的扩展包，并将aws_*database_engine_name*_ext架构添加到目标树中。接下来，将转换后的代码和扩展包架构 AWS SCT 应用于目标数据仓库。

   当您将 Amazon Redshift 和组合使用 AWS Glue 作为目标数据库平台时， AWS SCT 会在扩展包中添加一个额外的架构。

使用 AWS SCT 扩展包中的 AWS Lambda 函数

AWS SCT 提供了一个扩展包，其中包含用于电子邮件、作业计划的 Lambda 函数以及托管在 Amazon EC2 上的数据库的其他功能。

使用 AWS Lambda 函数来模拟数据库功能

在某些情况下，数据库功能无法转换为等效的 Amazon RDS 功能。例如，Oracle 发送使用 UTL_SMTP 的电子邮件调用，Microsoft SQL Server 可以使用作业计划程序。如果您在 Amazon EC2 上托管和自行管理数据库，则可以通过使用 AWS 服务代替这些功能来模拟这些功能。

AWS SCT 扩展包向导可帮助您安装、创建和配置 Lambda 函数，以模拟电子邮件、作业计划和其他功能。

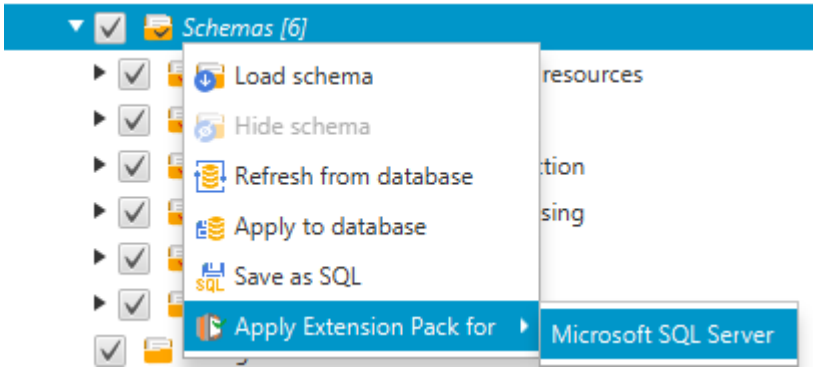
应用扩展包支持 Lambda 函数

您可以使用扩展包向导或在将转换后的代码应用于目标数据库时应用扩展包支持 Lambda 函数。

> **Important**
>
> 只有在 Amazon EC2 上安装和自行管理的数据库，才支持 AWS 服务仿真功能。如果目标数据库位于 Amazon RDS 数据库实例上，请勿安装服务模拟功能。

使用扩展包向导应用扩展包

1. 在目标数据库树中 AWS Schema Conversion Tool，打开上下文（右键单击）菜单，选择 A pply extension pack for，然后选择您的源数据库平台。



   扩展包向导随即出现。

2. 查看 Welcome 页面，然后选择 Next。
3. 在 AWS 配置文件页面上，执行以下操作。

   - 如果您只要重新安装扩展包架构，则选择 Skip this step for now，然后选择 Next。
   - 如果您正在安装 AWS 服务，请提供用于连接到您的服务的凭据 AWS 账户。如果您 AWS CLI 安装了 AWS CLI 证书，则可以使用您的凭证。您也可以使用之前存储在全局应用程序设置的配置文件中且与项目关联的凭证。如有必要，请选择 Navigate to Project Settings，将一个不同的配置文件与项目关联。如有必要，请选择 Global Settings 以创建新的配置文件。有关更多信息，请参阅 将 AWS 服务配置文件存储在 AWS SCT 中。

4. 在 Email Sending Service 页面上，执行以下操作：

   - 如果您只要重新安装扩展包架构，则选择 Skip this step for now，然后选择 Next。
   - 如果您正在安装 AWS 服务并且已有 Lambda 函数，则可以提供该函数。否则，向导将为您创建该函数。完成后，选择 Next 。

5. 在 Job Emulation Service 页面上，执行以下操作：

   - 如果您只要重新安装扩展包架构，则选择 Skip this step for now，然后选择 Next。
   - 如果您正在安装 AWS 服务并且已有 Lambda 函数，则可以提供该函数。否则，向导将为您创建该函数。完成后，选择 Next 。

6. 在函数模拟页面上，选择创建扩展包。此时显示包含扩展包操作状态的消息。

   完成后，选择 Finish。

> **Note**
>
> 要更新扩展包并覆盖旧的扩展包组件，请确保使用最新版本的 AWS SCT。有关更多信息，请参阅 安装、验证和更新 AWS SCT。

为 AWS SCT 扩展包配置函数

扩展包包含在使用前必须配置的功能。该常量CONVERSION_LANG定义了补丁包使用的语言。这些功能适用于英语和德语。

要将语言设置为英语或德语，请在函数代码中进行以下更改。找到以下常量声明：

```
CONVERSION_LANG CONSTANT VARCHAR := '';
```

CONVERSION_LANG要设置为英语，请将该行更改为以下内容：

```
CONVERSION_LANG CONSTANT VARCHAR := 'English';
```

CONVERSION_LANG要设置为英语，请将该行更改为以下内容：

```
CONVERSION_LANG CONSTANT VARCHAR := 'Deutsch';
```

为以下功能设置此设置：

- aws_sqlserver_ext.conv_datetime_to_string
- aws_sqlserver_ext.conv_date_to_string
- aws_sqlserver_ext.conv_string_to_date
- aws_sqlserver_ext.conv_string_to_datetime
- aws_sqlserver_ext.conv_string_to_datetime
- aws_sqlserver_ext.parse_to_date
- aws_sqlserver_ext.parse_to_datetime
- aws_sqlserver_ext.parse_to_time

AWS SCT 的最佳实践

查找关于使用 AWS Schema Conversion Tool (AWS SCT) 的最佳实践和选项信息。

配置额外的内存

为了转换大型数据库架构，例如含有 3500 个存储过程的数据库，您可以配置供 AWS Schema Conversion Tool 使用的内存容量。

修改 AWS SCT 使用的内存量

1. 在设置菜单上，选择全局设置，然后选择 JVM 选项。
2. 选择编辑配置文件，然后选择文本编辑器来打开配置文件。
3. 编辑 `JavaOptions` 部分以设置最小和最大可用内存。以下示例将最小值设置为 4 GB，最大值设置为 40 GB。

   ```
   [JavaOptions]
   -Xmx40960M
   -Xms4096M
   ```

   建议您将最小可用内存设置为至少 4 GB。
4. 保存配置文件，选择确定，然后重新启动 AWS SCT 以应用更改。

配置默认项目文件夹

AWS SCT 使用项目文件夹存储项目文件、保存评估报告和存储转换后的代码。默认情况下，AWS SCT 将所有文件存储在应用程序文件夹中。您可以将其他文件夹指定为默认项目文件夹。

更改默认项目文件夹

1. 在设置菜单上，选择全局设置，然后选择文件路径。
2. 在默认项目文件路径中，输入默认项目文件夹的路径。
3. 选择 Apply，然后选择 OK。

提高数据迁移速度

要迁移大型数据集，例如一组数据超过 1 TB 的表，可能需要提高迁移速度。使用数据提取代理时，数据迁移的速度取决于各种因素。这些因素包括目标 Amazon Redshift 集群中的切片数量、迁移任务中区块文件的大小、运行数据提取代理的 PC 上的可用 RAM 等。

为了提高数据迁移速度，建议使用生产数据的小数据集运行多个测试迁移会话。此外，建议您在装有至少 500 GB 的固态硬盘的 PC 上运行数据提取代理。在这些测试会话中，更改不同的迁移参数，监控磁盘利用率，找出可确保最大数据迁移速度的配置。然后，使用此配置迁移整个数据集。

增加日志记录信息

您可以增加转换数据库、脚本和应用程序 SQL 时由 AWS SCT 生成的日志记录信息。尽管增加日志记录信息可能会减慢转换速度，但这些更改可以帮助您在出现错误时向 AWS Support 提供可靠的信息。

AWS SCT 将日志存储在本地环境中。您可以查看这些日志文件并与 AWS Support 或 AWS SCT 开发人员共享以进行故障排除。

更改日志记录设置

1. 打开设置菜单，选择全局设置，然后选择日志记录。
2. 在日志文件夹路径中，输入用于存储用户界面日志的文件夹。
3. 在控制台日志文件夹路径中，输入用于存储 AWS SCT 命令行界面（CLI）日志的文件夹。
4. 在最大日志文件大小（MB）中，输入单个日志文件的大小（以 MB 为单位）。在文件达到此限制后，AWS SCT 创建一个新的日志文件。
5. 在最大日志文件数中，输入要存储的日志文件的数量。文件夹中的日志文件数量达到此限制后，AWS SCT 会删除最旧的日志文件。
6. 在提取器日志下载路径中，输入用于存储数据提取代理日志的文件夹。
7. 对于 Cassandra 提取器日志路径，请输入用于存储 Apache Cassandra 数据提取代理日志的文件夹。
8. 选择加载前询问路径，确保每次使用数据提取代理时 AWS SCT 都会询问日志的存储位置。
9. 对于调试模式，请选择 True。当标准 AWS SCT 日志不包含任何问题时，使用此选项可以记录其他信息。
10. 选择关键应用程序模块以增加日志记录信息。您可以增加以下应用程序模块的日志记录信息：

    - 一般性问题

- 加载程序
- 解析器
- 打印机
- 解析程序
- 遥测
- 转换器
- 类型映射
- 用户界面
- 控制器
- 比较架构
- 克隆数据中心
- 应用程序分析器

对于上述每个应用程序模块，请选择下列日志记录级别之一：

- 跟踪：最详细的信息。
- 调试：有关系统流量的详细信息。
- 信息：运行时事件，例如启动或关闭。
- 警告：使用已弃用的 API、API 使用不当、其他不良或意外的运行时情况。
- 错误：运行时错误或意外情况。
- 严重：导致应用程序关闭的错误。
- 强制性：可能的最高错误级别。

默认情况下，在打开调试模式后，AWS SCT 会为所有应用程序模块设置信息日志记录级别。

例如，为了帮助解决转换过程中的关键问题区域，请将解析器、类型映射和用户界面设置为跟踪。

如果对于流式传输日志的文件系统而言，信息变得过于详细，请切换到有足够空间捕获日志的位置。

要将日志传输到 AWS Support，请转到存储日志的目录，然后将所有文件压缩成易于管理的单个.zip 文件。然后上传带支持案例的 .zip 文件。当初始分析完成并且持续开发恢复时，将调试模式恢复为 false 以清除详细日志记录。然后提高转换速度。

> **Tip**
>
> 要管理日志大小并简化报告问题，请在成功转换后删除日志或将其移至其他位置。执行此任务可确保仅将相关的错误和信息传输给 AWS Support，并防止填充日志文件系统。

排查 AWS SCT 问题

您可以在下面找到有关解决 AWS Schema Conversion Tool (AWS SCT) 方面的问题的信息。

无法从 Oracle 源数据库加载对象

当您尝试从 Oracle 数据库加载架构时，您可能会遇到以下错误之一。

```
Cannot load objects tree.
```

```
ORA-00942: table or view does not exist
```

引发这些错误的原因是，您用其 ID 连接到 Oracle 数据库的用户没有 AWS SCT 所要求的读取架构的足够权限。

您可以向用户授予 `select_catalog_role` 权限以及访问数据库中任何词典的权限，从而解决此问题。这些权限提供对 AWS SCT 所需的视图和系统表的只读访问。以下示例会创建一个名为 `min_privs` 的用户 ID，并向持有此 ID 的用户授予转换 Oracle 源数据库中架构所需的最小权限。

```
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;
```

评估报告警告消息

要评估转换为其他数据库引擎的复杂性，AWS SCT 需要访问源数据库中的对象。如果在扫描过程中 AWS SCT 遇到问题而无法进行评估，则会发出警告消息。此消息表示总体转换百分比降低。以下是在扫描过程中 AWS SCT 可能遇到问题的原因：

- 您的数据库用户没有对所有需要的对象的访问权限。有关 AWS SCT 所需的安全权限和数据库权限的更多信息，请参阅本指南中相应的源数据库部分 AWS SCT 的源。
- 数据库中不再存在架构中引用的对象。为了帮助解决问题，您可以使用 SYSDBA 权限进行连接并检查数据库中是否存在该对象。
- SCT 正在尝试评估加密的对象。

AWS SCT CLI 参考

本节介绍如何开始使用 AWS SCT 命令行界面 (CLI)。此外，本节还提供有关关键命令和使用模式的信息。有关 AWS SCT CLI 命令的完整参考，请参见参考材料。

主题

使用 AWS SCT 命令行界面的先决条件

下载并安装最新版本的 Amazon Corretto 11。有关更多信息，请参阅《Amazon Corretto 11 用户指南》中适用于 Amazon Corretto 11 的下载内容。

下载并安装最新版本的 AWS SCT。有关更多信息，请参阅 正在安装 AWS SCT。

AWS SCT CLI 交互模式

可以在交互模式下使用 AWS SCT 命令行界面。在此模式下，您可以将命令逐个输入控制台。您可以使用此交互模式了解有关 CLI 命令的更多信息或下载最常用的 CLI 场景。

要转换源数据库架构 AWS SCT，请运行序列操作：创建新项目、连接到源数据库和目标数据库、创建映射规则以及转换数据库对象。由于此工作流程可能很复杂，因此我们建议在 AWS SCT CLI 模式下使用脚本。有关更多信息，请参阅 脚本模式。

您可以从 AWS SCT 安装路径的app文件夹中运行 AWS SCT CLI 命令。在 Windows 中，默认安装路径为 `C:\Program Files\AWS Schema Conversion Tool\`。确保此文件夹中包含 `AWSSchemaConversionToolBatch.jar` 文件。

要进入 AWS SCT CLI 交互模式，请在完成先决条件后使用以下命令。

```
java -jar AWSSchemaConversionToolBatch.jar -type interactive
```

现在，您可以运行 AWS SCT CLI 命令了。确保新行以 / 结束命令。此外，请确保在命令参数值前后使用直单引号（ ' ）。

> **Note**
>
> 如果前面的命令返回 Unexpected error，请执行以下步骤：
>
> ```
> java -Djdk.jar.maxSignatureFileSize=20000000 -jar
>  AWSSchemaConversionToolBatch.jar
> ```

要查看 AWS SCT CLI 交互模式下的可用命令列表，请运行以下命令。

```
help
/
```

要查看有关 AWS SCT CLI 命令的信息，请使用以下命令。

```
help -command: 'command_name'
/
```

在前面的示例中，将 *command_name* 替换为命令名称。

要查看有关 AWS SCT CLI 命令参数的信息，请使用以下命令。

```
help -command: 'command_name' -parameters: 'parameters_list'
/
```

在前面的示例中，将 *command_name* 替换为命令名称。然后，将 *parameters_list* 替换为用逗号分隔的参数名称列表。

要在 AWS SCT CLI 交互模式下从文件运行脚本，请使用以下命令。

```
ExecuteFile -file: 'file_path'
/
```

在前面的示例中，将 *file_path* 替换为脚本文件路径。确保文件有 .scts 扩展名。

要退出 AWS SCT CLI 交互模式，请运行quit命令。

示例

以下示例显示有关 Convert 命令的信息：

```
help -command: 'Convert'
/
```

以下示例显示有关 Convert 命令两个参数的信息。

```
help -command: 'Convert' -parameters: 'filter, treePath'
/
```

获取 AWS SCT CLI 场景

要获取最常用的 AWS SCT 场景，你可以使用GetCliScenario命令。您可以在交互模式下运行此命令，然后编辑下载的模板。在脚本模式下使用编辑的文件。

GetCliScenario 命令将所选模板或所有可用模板保存到指定目录。模板包含运行脚本的完整命令集。请务必编辑这些模板中的文件路径、数据库凭证、对象名和其他数据。另外，请确保删除不使用的命令，并在需要时向脚本中添加新命令。

要运行 GetCliScenario 命令，请完成先决条件并进入 AWS SCT CLI 交互模式。有关更多信息，请参阅 交互模式。

接下来，使用以下语法运行 GetCliScenario 命令并获取 AWS SCT 场景。

```
GetCliScenario -type: 'template_type' -directory: 'file_path'
/
```

在前面的示例中，将 *template_type* 替换为下表中的一种模板类型。接下来，将 *file_path* 替换为要下载脚本的文件夹路径。确保它 AWS SCT 可以在不请求管理员权限的情况下访问此文件夹。此外，请确保在命令参数值前后使用直单引号（ ' ）。

要下载所有 AWS SCT CLI 模板，请在不带-type选项的情况下运行前面的命令。

下表包括您可以下载的 AWS SCT CLI 模板类型。对于每个模板，该表都包含文件名和可使用该脚本运行的操作的描述。

模板类型	文件名	描述
BTEQ ScriptConversion	`BTEQScriptConversionTemplate.scts`	将 Teradata 基本 Teradata 查询 (BTEQ)、、 FastExport和FastLoad脚本转换为 Amazon Redshift RSQL MultiLoad 。有关更多信息，请参阅 转换 ETL 过程。
ConversionApply	`ConversionTemplate.scts`	转换源数据库架构并将转换后的代码应用于目标数据库。或者，将转换后的代码另存为 SQL 脚本，并保存评估报告。有关更多信息，请参阅 转换数据库架构。
GenericAppConversion	`GenericApplicationConversionTemplate.scts`	使用通用 AWS SCT 应用程序转换器转换嵌入到应用程序中的 SQL 代码。有关更多信息，请参阅 转换应用程序中的 SQL 代码。
HadoopMigration	`HadoopMigrationTemplate.scts`	将您的本地 Hadoop 集群迁移到 Amazon EMR。有关更多信息，请参阅 使用 Apache Hadoop 作为 AWS SCT 的源。
HadoopResumeMigration	`HadoopResumeMigrationTemplate.scts`	恢复中断的从本地 Hadoop 集群到 Amazon EMR 的迁移。有关更多信息，请参阅 使用 Apache Hadoop 作为 AWS SCT 的源。
Informatica	`InformaticaConversionTemplate.scts`	转换嵌入到 Informatica 提取、转换和加载（ETL）脚本中的 SQL 代码。在 ETL 脚本中配置与源数据库和目标数据库的连

模板类型	文件名	描述
		接，并在转换后保存转换后的脚本。有关更多信息，请参阅转换 Informatica ETL 脚本。
LanguageSpecificAppConversion	`LanguageSpecificAppConversionTemplate.scts`	使用 AWS SCT 应用程序转换器转换嵌入到 C#、C++、Java 和 Pro*C 应用程序中的 SQL 代码。有关更多信息，请参阅转换应用程序 SQL。
OozieConversion	`OozieConversionTemplate.scts`	将你的 Apache Oozie 工作流程转换为。 AWS Step Functions有关更多信息，请参阅 使用 Apache Oozie 作为 AWS SCT 的源 。
RedshiftAgent	`DWHDataMigrationTemplate.scts`	转换源数据仓库架构，并将转换后的代码应用于目标 Amazon Redshift 数据库。然后，注册数据提取代理，创建并启动数据迁移任务。有关更多信息，请参阅 从数据仓库迁移到亚马逊 Redshift。
ReportCreation	`ReportCreationTemplate.scts`	创建多个源数据库架构的数据库迁移报告。然后将此报告另存为 PDF 文件或 CSV 文件。有关更多信息，请参阅 迁移评估报告。
SQL ScriptConversion	`SQLScriptConversionTemplate.scts`	将 SQL*Plus 或 TSQL 脚本转换为 PL/SQL 并保存转换后的脚本。此外，保存评估报告。

下载 AWS SCT CLI 模板后，使用文本编辑器将脚本配置为在源数据库和目标数据库上运行。接下来，使用 AWS SCT CLI 脚本模式运行脚本。有关更多信息，请参阅 AWS SCT CLI 脚本模式。

示例

以下示例将所有模板下载到 C:\SCT\Templates 文件夹。

```
GetCliScenario -directory: 'C:\SCT\Templates'
/
```

以下示例将 ConversionApply 操作模板下载到 C:\SCT\Templates 文件夹。

```
GetCliScenario -type: 'ConversionApply' -directory: 'C:\SCT\Templates'
/
```

编辑 AWS SCT CLI 场景

下载场景模板后，将其配置为可在数据库上运行的工作脚本。

对于所有模板，请务必提供源数据库和目标数据库驱动程序的路径。有关更多信息，请参阅 下载所需的数据库驱动程序。

确保包含源数据库和目标数据库的数据库凭证。此外，请务必设置映射规则以描述转换项目的源目标对。有关更多信息，请参阅 创建映射规则。

接下来，配置要运行的操作的范围。您可以删除不使用的命令或向脚本中添加新命令。

例如，假设您计划将源 Oracle 数据库中的所有架构转换为 PostgreSQL。然后，您计划将数据库迁移评估报告另存为 PDF，并将转换后的代码应用于目标数据库。在这种情况下，您可以使用 ConversionApply 操作模板。使用以下步骤编辑您的 AWS SCT CLI 模板。

编辑ConversionApply操作的 AWS SCT CLI 模板

1. 打开下载的 ConversionTemplate.scts。有关更多信息，请参阅 示例。
2. 从脚本中移除CreateFilter、转换 ApplyToTarget -f ilt SaveTarget er、-filter LbyStatement、SaveTarget SQL 和 SaveReportCSV 操作。
3. 对于SetGlobalSettings操作中的 oracle_driver_file，请输入 Oracle 驱动程序的路径。然后，在 postgresql_driver_file 中，输入 PostgreSQL 驱动程序的路径。

   如果使用其他数据库引擎，请使用相应的设置名称。有关可在SetGlobalSettings操作中设置的全局设置的完整列表，请参阅中的全局设置矩阵参考材料。

4. （可选）对于 CreateProject，请输入您的项目名称和本地项目文件的位置。如果您选择继续使用默认值，请确保无需请求管理员权限 AWS SCT 即可在 C:\temp 文件夹中创建文件。
5. 对于 AddSource，请输入源数据库服务器的 IP 地址。此外，输入用于连接到源数据库服务器的用户名、密码和端口。
6. 对于 AddTarget，输入目标数据库服务器的 IP 地址。此外，输入用于连接到目标数据库服务器的用户名、密码和端口。
7. （可选）对于 AddServerMapping，输入要添加到映射规则的源和目标数据库对象。您可以使用 sourceTreePath 和 targetTreePath 参数指定数据库对象的路径。或者，您可以使用 sourceNamePath 和 targetNamePath 指定数据库对象的名称。有关更多信息，请参阅 参考材料 中的服务器映射命令。

   该AddServerMapping操作的默认值将所有源架构映射到您的目标数据库。
8. 保存文件，然后使用脚本模式运行文件。有关更多信息，请参阅 脚本模式。

AWS SCT CLI 脚本模式

创建 AWS SCT CLI 脚本或编辑模板后，您可以使用RunSCTBatch命令运行该脚本。确保将带 CLI 脚本的文件另存为 .scts 扩展文件。

您可以从 AWS SCT 安装路径的app文件夹中运行 AWS SCT CLI 脚本。在 Windows 中，默认安装路径为 C:\Program Files\AWS Schema Conversion Tool\。确保此文件夹包含 RunSCTBatch.cmd 或 RunSCTBatch.sh 文件。此外，此文件夹还应包含 AWSSchemaConversionToolBatch.jar 文件。

或者，也可以在操作系统的 PATH 环境变量中添加 RunSCTBatch 文件路径。更新PATH环境变量后，可以从任何文件夹运行 AWS SCT CLI 脚本。

要运行 AWS SCT CLI 脚本，请在 Windows 中使用以下命令。

```
RunSCTBatch.cmd --pathtoscts "file_path"
```

在前面的示例中，将 *file_path* 替换为脚本文件路径。

要运行 AWS SCT CLI 脚本，请在 Linux 中使用以下命令。

```
RunSCTBatch.sh --pathtoscts "file_path"
```

在前面的示例中，将 *file_path* 替换为脚本文件路径。

您可以在此命令中提供可选参数，例如数据库凭证、控制台输出中的详细信息级别等。有关更多信息，请下载 AWS SCT 命令行界面参考，网址为参考材料。

示例

以下示例运行 C:\SCT\Templates 文件夹中的 ConversionTemplate.scts 脚本。您可以在 Windows 中使用这个示例。

```
RunSCTBatch.cmd --pathtoscts "C:\SCT\Templates\ConversionTemplate.scts"
```

以下示例在 /home/user/SCT/Templates 目录中运行 ConversionTemplate.scts 脚本。您可以在 Linux 中使用这个例子。

```
RunSCTBatch.sh --pathtoscts "/home/user/SCT/Templates/ConversionTemplate.scts"
```

AWS SCT CLI 参考资料

您可以在以下指南中找到有关 AWS Schema Conversion Tool 命令行界面 (CLI) 的参考资料：AWS Schema Conversion Tool CLI 参考。

的发行说明 AWS SCT

本节包含从 1.0.6 AWS SCT 40 版本开始的版本说明。

AWS SCT 版本 676 的发布说明

来源	目标	新增功能、增强或修复	S AWS DMS chema Conversion Tool (SCT) 中的可用性	AWS DMS 架构转换的可用性
Oracle	PostgreSQL/Aurora PostgreSQL	以下函数的新内置函数仿真： • `SYS.UTL_RAW.BIT_AND(RAW,RAW)` • `XDB.DBMS_XSLPROCESSOR.CLOB2FILE(CLOB)` • `XDB.DBMS_XSLPROCESSOR.READ2CLOB(VARCHAR2)` • `SYS.UTL_RAW.BIT_OR(RAW,RAW)` • `SYS.UTL_RAW.BIT_COMPLEMENT(RAW)`	否	是
MS SQL Server	亚马逊 RDS SQL 服务器	已从 PDF 报告中删除`Database Mail not supported` 消息	是	是
Oracle	PostgreSQL/Aurora	实现了分区表的约束条件转换。	是	是

来源	目标	新增功能、增强或修复	S AWS DMS chema Conversion Tool (SCT) 中的可用性	AWS DMS 架构转换的可用性
	Postgre L			
Oracle	MySQL	审查 AI-602 在表格转换中的适用性	是	是
MS SQL Server	Postgre L/ Aurora Postgre L	现在支持 PostgreSQL 15.x 中的MERGE声明	是	是
全部	全部	实现的 JDBC 连接：高级属性	是	不支持
All	全部	CLI：修复了PrintOLAPTaskStatus 命令失败的问题	是	不支持
Teradat	Amazor Redshif	实现了 Teradata 风格的数据类型转换。	是	不支持
Teradat	Amazor Redshif	修复了 SQL/BTEQ 中MERGE转换不正确的问题。	是	不支持
Teradat	Amazor Redshif	实现了 Teradata 风格的数据类型转换。	是	不支持
Teradat	Amazor Redshif	实现了LEAD/LAG函数转换。	是	不支持
Teradat	Amazor Redshif	修复了错误AI-9996 Transformer error occurred in statement 。	是	不支持
Teradat	Amazor Redshif	修复了错误AI-9996 Transformer error in selectItem 。	是	不支持

来源	目标	新增功能、增强或修复	S AWS DMS chema Conversion Tool (SCT) 中的可用性	AWS DMS 架构转换的可用性
Teradat	Amazor Redshif	实现了部分存储过程的转换：XbiDQM.Sp CmprsnDly	是	不支持
Teradat	Amazor Redshif	实现了带别名的UNPIVOT语句。	是	不支持
Teradat	Amazor Redshif	使用多个源Delete表实现了语句。	是	不支持
Teradat	Amazor Redshif	修复AI-9996 Transformer error occurred in functionCallExpres sion .	是	不支持
Teradat	Amazor Redshif	已实现NORMALIZE 子句转换。	是	不支持
Teradat	Amazor Redshif	修复了带有子查询的DELETE语句中的错误转换。	是	不支持
Teradat	Amazor Redshif	修复了错误AI-9996 Transformer error occurred in tableOper atorSource 。	是	不支持
Teradat	Amazor Redshif	修复了错误AI-9996 Transformer error occurred in additiveE xpression 。	是	不支持
Teradat	Amazor Redshif	实现了 DBC 系统对象转换。	是	不支持
Teradat	Amazor Redshif	实现了使用隐式联接谓词更新的解决方法。	是	不支持

来源	目标	新增功能、增强或修复	S AWS DMS chema Conversion Tool (SCT) 中的可用性	AWS DMS 架构转换的可用性
Netezza	Amazon Redshift	修复了CREATE MATERIALIZED VIEW 语句转换错误。	是	不支持
db2luw	PostgreSQL/ Aurora PostgreSQL	JDBC 扩展选项连接：添加了其他连接选项。	是	不支持
db2luw	PostgreSQL/ Aurora PostgreSQL	在 PostgreSQL MERGE 15.x 中增加了对语句的支持	是	不支持
db2luw	PostgreSQL/ Aurora PostgreSQL	已实现GLOBAL TEMPORARY TABLE转换。	是	不支持
db2luw	PostgreSQL/ Aurora PostgreSQL	已实现USER DEFINED TYPES转换。	是	不支持
db2luw	MySQL	已实现GLOBAL TEMPORARY TABLE转换。	是	不支持
db2luw	MySQL	已实现USER DEFINED TYPES转换。	是	不支持

来源	目标	新增功能、增强或修复	S AWS DMS chema Conversion Tool (SCT) 中的可用性	AWS DMS 架构转换的可用性
db2luw	MySQL	已实现USER DEFINED FUNCTIONS 转换。	是	不支持
db2luw	MariaDB	已实现GLOBAL TEMPORARY TABLE转换。	是	不支持
db2luw	MariaDB	已实现USER DEFINED TYPES转换。	是	不支持
Sybase	全部	增加了对 Kerberos 身份验证的支持	是	不支持
db2luw	PostgreSQL/ Aurora PostgreSQL	增加了对目标的多版本转换的支持	是	不支持
Azure SQL/ Microsoft SQL Server	PostgreSQL/ Aurora PostgreSQL	增加了对目标的多版本转换的支持	是	不支持
db2luw	PostgreSQL/ Aurora PostgreSQL	在 PostgreSQL MERGE 15.x 中增加了对语句的支持。	是	不支持
Teradata	Amazon Redshift	修复了不支持的函数更改转换。	是	不支持

来源	目标	新增功能、增强或修复	S AWS DMS chema Conversion Tool (SCT) 中的可用性	AWS DMS 架构转换的可用性
All	Amazor Redshif	数据提取器：实现了按索引列进行分区。	是	不支持

B AWS SCT uild 675 的发布说明

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
Cassandr	DynamoD	修复了在目标数据中心安装 Cassandra 会失败的错误。	否
DB2 LUW	PostgreS(L	动态 SQL：准备语句：不使用动态 SQL 进行解析和转换。	否
DB2 LUW	PostgreS(L	增加了对特殊寄存器的支持。	否
DB2 LUW	PostgreS(L	扩展包更新	否
Hadoop	Amazon EMR	增加了对通过 rsa-sha2 协议连接到 Hadoop 集群的支持。	否
Microsoft SQL Server	Amazon Redshift	修复了 JDBC 驱动程序在未配置的情况下仍强制执行 TLS 的问题。	否
Netezza	Amazon Redshift	增加了对物化视图转换的支持。	否

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
Oracle	Amazon Redshift	在 Amazon Redshift 中增加了对递归查询的支持。	是
Oracle	PostgreSQL、Aurora PostgreSQL	修复了 NUMBER 数据类型转换不正确的问题。	是
Oracle	Amazon Redshift	数据迁移。甲骨文自动分区。为表片段值添加了过期时间。到期时间为 72 小时。过期时，在创建数据迁移任务时会重建数据片段。	否
Oracle	Amazon Redshift	SCT 数据提取器：更改了将数据上传到 Amazon Redshift 的方法。默认情况下，提取器不创建暂存表。取而代之的是，在所有数据文件都存入 Amazon S3 存储桶后，提取器使用单个 COPY 命令将它们复制到目标表。	否
Oracle	Amazon Redshift	添加了将 RAW 数据类型迁移到 VARBYTE 列中。	否
Oracle	PostgreSQL、Aurora PostgreSQL	多版本转换	否
Oracle	PostgreSQL	在 PostgreSQL 15.x 中增加了对合并语句的支持。	是
Oracle	PostgreSQL	在 PostgreSQL 15.x 中增加了对新正则表达式函数的支持。	是
Oracle	PostgreSQL、Aurora PostgreSQL	ON CONFLICT DO UPDATE 语句转换时不排除别名。	是

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
Teradata	Amazon Redshift	增加了对 LEAD/LAG 函数的转换支持。	否
Teradata	Amazon Redshift	增强了数据类型转换功能，明确指示了数据格式。	否
Teradata	Amazon Redshift	改进了时间/时间戳表达式中 AT '时区'子句的转换。	否
Teradata	Amazon Redshift	AI-9996 在转换过程中使用 MERGE 语句。	否

AWS SCT 版本 674 的发布说明

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
全部	全部	各种错误修复和性能改进	部分（仅适用于支持的源和目标对）
Azure SQL/ Microsoft SQL Server	Amazon Redshift	删除了在架构评估/转换期间出现的误导用户的消息“AI 18066：无法转换架构名称”	否
Azure SQL/ Microsoft SQL Server	Amazon RDS for MySQL / Amazon Aurora MySQL	在未分配返回代码的情况下转换过程不正确	部分（架构转换目前不支持将 Azure SQL 作为源）

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	修复了某些情况下 FOR XML PATH 子句转换时出现的 AI9997	部分（架构转换目前不支持将 Azure SQL 作为源）
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	在过程/函数体中，值四舍五入，保留原始小数位数。	部分（架构转换目前不支持将 Azure SQL 作为源）
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	对 EXECUTE 语句的转换进行了各种改进	部分（架构转换目前不支持将 Azure SQL 作为源）

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
Azure SQL/ Microsoft SQL Server/ Azure Synapse	Amazon Redshift	改进了以下语句和模式的转换： • EXCEPTION BLOCK • AUTOCOMMIT • NONATOMIC • GROUPING SET • CUBE • ROLLUP	否
DB2 LUW	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	对元数据加载 SQL 查询进行了各种修复。	否
DB2 LUW	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	触发器上预计不会出现 AI 9996	否

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	ROWNUMBER 分析函数	否
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	十六进制字符串常量支持	否
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	对元数据加载 SQL 查询进行了各种修复。	否

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	NEXT VALUE FOR 序列引用支持	否
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	GET DIAGNOSTICS 语句 DB2_NUMBER_ROWS 选项支持	否
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	GET DIAGNOSTICS 多语句	否

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	修复了 FOR 语句转换中的错误。	否
Oracle	Amazon RDS for MySQL / Amazon Aurora MySQL	修复了未定义包函数的参数节点时出现的错误。	是
Oracle	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	修复了扩展包函数 AWS_ORACLE_EXT.NEXT_DAY 中的错误	是
Oracle	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	修复了 Oracle 外联接中转换“（+）”的各种错误	是

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
Oracle		支持 Kerberos 身份验证	否
SAP ASE	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	修复了在 UPDATE 语句 FROM 子句中转换多个标识符时出现的错误	否
SAP ASE	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	修复了多行注释和语句转换错误	否
SAP ASE		添加了连接时对 ENCRYPT_PASSWORD 参数的支持	否
Teradata	Amazon Redshift	改进了具有指定架构名称的 VOLATILE 表的转换	否
Teradata	Amazon Redshift	复杂 CTE 中的 WHERE 子句转换不正确	否
Teradata	Amazon Redshift	增加了在使用 SCT 数据提取代理迁移数据时对 INTERVAL 数据类型的支持。	否

来源	目标	新增功能、增强或修复	AWS DMS 架构转换的可用性
Teradata BTEQ 脚本	Amazon Redshift RSQL 脚本	在 BTEQ 执行的过程中输出参数转换不正确	否

AWS SCT 版本 673 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	常规性能改进和错误修复。
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	修复了错误的函数调用转换
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	实现了 FOR XML 子句的转换
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	使用错误别名转换 FOR XML 子句。
Azure SQL/	Aurora PostgreSQ	修复了 AWS SCT 无法转换运行带有过程参数的字符串的语 EXECUTE句的错误。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	L/ Amazon RDS PostgreSQL	
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	改进了使用内部联接的 UPDATE 语句的转换。
Azure Synapse	Amazon Redshift	修复了 OBJECT_ID 内置函数转换不正确的问题。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	实现了以下语句和对象的转换： • DECLARE TEMPORARY TABLE statement • DROP TABLE statement • 分区表上 PK 和 UNIQUE 约束 • TIMESTAMPDIFF 函数 • TO_DATE 函数 • EBCDIC_STR 函数 • VARCHAR_FORMAT 函数
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	修复了基于函数的索引在转换后跳过函数的错误。

来源	目标	新增功能、增强或修复
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	修复了转换后 REPEAT 语句以 AI 9996 结尾的错误
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	修复了 FINAL TABLE 子句以 9996 结尾的错误。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	LOADER | 引用约束中的分区键。 AWS SCT 现在可以将分区表中的主键和唯一约束转换为二级索引。
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	PostgreSQL.VARCHAR_FORMAT 函数支持
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	在 CreateTransformationRule 和 ModifyTransformationRule SCT CLI 命令中实现了排序规则更改。

来源	目标	新增功能、增强或修复
Greenplum	Amazon Redshift	修复了转换后不正确调用存储过程的错误
Hadoop	Amazon EMR	增加了对使用 rsa-sha2 协议连接到 Hadoop 集群的支持。
Hadoop	Amazon EMR	增加了对带有非 Glue Hive 元存储的 Amazon EMR 的支持，
Oracle	Amazon Redshift	修复了 PRIOR 列不在 SELECT 列表中的递归查询转换不正确的错误。
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	实现了返回关联数组元素
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	使用方括号修复 `UNPIVOT` 中意外出现的 AI 9996
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	使用 `UNION ALL` 修复 `UNPIVOT` 中意外出现的 AI 9996

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	Number 数据类型转换的改进
Oracle	Amazon Redshift 数据提取器	支持 Oracle 表的自动分区。针对创建迁移任务进行了优化。
Teradata	Amazon Redshift	实现 EXCEPTION BLOCK 语句的转换
Teradata	Amazon Redshift	支持将 ALL、ANY、和 SOME 谓词转换为 Amazon Redshift。
Teradata	Amazon Redshift	添加了对 QUALIFY 谓词的原生支持。
Teradata	Amazon Redshift	改进了以下内容的转换： • 递归查询 • GROUPING SET • CUBE • ROLLUP • 带有隐式连接的 UPDATE 语句
OLAP 的源	Amazon Redshift 数据提取器	实现了 CLI 命令以“停止/恢复” Amazon Redshift 数据提取器任务。
OLAP 的源	Amazon Redshift 数据提取器	增加了在配置迁移任务期间选择需要迁移的表列的功能。

AWS SCT 版本 672 的发布说明

来源	目标	新增功能、增强或修复
全部	Amazon RDS for PostgreSQL	支持将 PostgreSQL 主要版本 15 作为迁移目标。
全部	Amazon Redshift	在`PrintTaskStatus` AWS SCT 命令行界面 (CLI) 中添加了一个新命令来显示数据迁移任务的状态。
全部	Amazon Redshift	改进了数据提取代理的配置流程。
全部	Amazon Redshift	修复了数据提取代理未显示有关子任务的信息的错误。
Apache Oozie	AWS Step Functions	在转换后的代码中添加了将状态机定义另存为脚本的选项。
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	实现了 `COALESCE`、`DATEADD`、`GETDATE` 和 `SUM` 函数的转换。
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了带 `JOIN` 和 `OUTPUT` 子句的 `UPDATE` 语句的转换。
Azure SQL 数据库	Aurora PostgreSQL	修复了 `SELECT TOP 1 WITH TIES` 语句转换过程中发生的错误。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	PostgreSQL	
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了在转换内置函数中的 FOR XML 子句时出现的多个问题。
Greenplum	Amazon Redshift	使用原生 Amazon Redshift EXCEPTION 块实现了 GET DIAGNOSTICS 和 RAISE 语句的转换。
Greenplum	Amazon Redshift	通过在转换后的代码中添加对 EXCEPTION 块的支持，改进了存储过程的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	修复了带有时间格式模板的 TO_CHAR 函数转换不正确的错误。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了嵌套表表达式的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了GOTO、MERGE、REPEAT 和SIGNAL 语句的转换。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了带有 BEFORE 和 AFTER 方向关键字的 FETCH 语句的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了 FINAL TABLE 和 OLD TABLE 表引用的转换。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了以下函数的转换。 • ADD_MONTHS • 使用字符数据类型的参数的 DAY • DAYOFWEEK • DAYS • DECODE • HOUR • LAST_DAY • LOCATE_IN_STRING • MICROSECOND • MINUTE • MONTH • ROUND • TIME • TIMESTAMP • TIMESTAMP_FORMAT • TRANSLATE • UNICODE_STR • XMLCAST • XMLELEMENT • XMLQUERY • XMLSERIALIZE • YEAR
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 JOIN 子句中子查询别名的转换。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 COALESCE 函数的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 EXPLICIT 索引的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了复合表达式中列名的转换，以解决转换过程中操作项 9997 意外出现的问题。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了主键和唯一约束的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 INSERT 语句中 XMLTABLE 语句的转换，以解决在转换过程中意外出现操作项 9996 的问题。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了在转换带有 SUBSTR 参数的函数时意外出现操作项 9996 的问题。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了在转换 CURRENT TIMESTAMP 特殊寄存器期间意外出现操作项 9996 的问题。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了在转换 MERGE 语句、不支持的语句和不支持的内置函数时意外出现操作项 9996 的问题。
Microsoft SQL Server	全部	增加了对将 Microsoft SQL Server 版本 2022 用作源的支持。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了使用字符串连接运算符的SELECT语句的转换。 AWS SCT 在转换后的代码中使用该STRING_AGG 函数。
Microsoft SQL Server	适用于 Aurora PostgreSQL 的 Babelfish	实现了对 Babelfish 功能配置文件新版本 3.1.0 的支持。此文件定义了特定 Babelfish 版本支持和不支持的 SQL 功能。

来源	目标	新增功能、增强或修复
Netezza	Amazon Redshift	解决了数据提取代理无法从指定的 CDC 点开始数据迁移的问题。
Oracle	全部	更新了作为源的 Oracle 数据库版本 19 的评估报告。
Oracle	Aurora PostgreSQL PostgreSQL	通过向 AWS SCT 扩展DBMS_OUTPUT 包中添加新函数来实现软件包的转换。
Oracle	Aurora PostgreSQL PostgreSQL	实现了使用关联数组作为自变量或参数的函数和过程的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了 SELECT 语句中 DISTINCT 子句的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了主键约束与表同名的表的转换。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	改进了使用第三个参数的 RAISE_APPLICATION_ERROR 过程的转换。
Oracle	Aurora PostgreSQL PostgreSQL	解决了迁移规则未自动将 NUMERIC 数据类型更改为 INTEGER（如果适用）的问题。
Oracle 数据仓库	Amazon Redshift	实现了对转换后的代码中原生 Amazon Redshift CONNECT BY 子句的支持。
Oracle 数据仓库	Amazon Redshift	通过自动为迁移范围内的每个表或分区添加子任务改进数据迁移。这种方法可以防止分区后插入的数据丢失。
Teradata	Amazon Redshift	实现了递归视图的转换。
Teradata	Amazon Redshift	通过添加对原生 Amazon Redshift AUTOCOMMIT 事务模式的支持，改进了使用 BTET 和 ANSI 事务模式的存储过程的转换。
Teradata	Amazon Redshift	通过在转换后的代码中添加 NONATOMIC 关键字，改进了使用 TERADATA 事务语义的存储过程的转换。
Teradata	Amazon Redshift RSQL	解决了转换后的代码包含 AWS 访问密钥 ID 和私有访问密钥的问题。

AWS SCT 版本 671 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	修复了在 Windows 中无权保存项目文件的错误。 AWS SCT
全部	全部	更新了以下 AWS SCT 命令行界面 (CLI) 模板。 • BTEQ ScriptConversion • ConversionApply • HadoopMigration • HadoopResume迁移 • Informatica 有关 AWS SCT CLI 模板的更多信息，请参阅获取 CLI 场景。
全部	Amazon Redshift	修复了 AWS SCT 未在命令行界面 (CLI) 中创建扩展包的错误。
全部	Amazon Redshift	解决了 AWS SCT 数据提取代理未在命令行界面 (CLI) 中使用 AWS Snowball 配置的问题。
Apache Oozie	AWS Step Functions	支持从 Apache Oozie 迁移到 AWS Step Functions 命令行界面 (CLI) 模式。将 Hadoop 工作负载迁移到 Amazon EMR 后，您现在可以将工作流程计划系统迁移到 AWS Cloud。有关更多信息，请参阅 将 Apache Oozie 转换为 AWS Step Functions。
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了表和别名出现的解析器错误。
Azure SQL 数据库	Aurora PostgreSQL	实现了 `INDEX ON` 子句的转换。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	PostgreSQL	
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了以下对象的转换，以避免出现意外操作项。 • 批处理语句 • 表达式列表 • 表别名 • 临时表 • 触发 • 用户变量
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了过程发生的解析错误。
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了在转换后的`OBJECT_ID` 函数代码中 AWS SCT 使用不正确的临时表名称的错误。

来源	目标	新增功能、增强或修复
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了在转换以下代码元素时意外出现操作项 9996 的问题。 • CONVERT 函数 • DATEADD 函数 • 内联函数中的 DELETE 语句 • IF 语句 • 对列执行的 INSERT 或 UPDATE 操作 • RETURN 语句 • 包含复杂查询或函数的 UPDATE 语句
BigQuery	Amazon Redshift	增加了对 BigQuery 作为多服务器评估过程来源的支持。有关更多信息，请参阅 创建多服务器评估报告。
Hadoop	Amazon EMR	更新了用于连接源数据库的支持的 Apache Hive JDBC 驱动程序的版本。有关更多信息，请参阅 下载所需的数据库驱动程序。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	增强了源元数据加载器，以确保 AWS SCT 加载源数据库对象，例如主键、隐式索引等。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	修复了隐式游标中的列发生的解析器错误。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了在转换后的代码中保留 DML 语句中列名、表达式和子句格式的功能。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了跨架构外键的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了 LENGTH 和 VARCHAR 函数的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了 LABEL ON 和 DECLARE CONDITION 语句的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了带 OPTIMIZE FOR 子句的 SELECT 语句的转换。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	通过为所有支持的数据类型添加默认值，改进了 CREATE TABLE 语句的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 INCREMENT BY 属性的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	通过添加将表分区排除在转换范围之外的功能，改进了分区表的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 INCLUDE 列主键定义的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 SUBSTRING 函数的转换。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 SET 和 DECLARE HANDLER FOR 语句的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了变量数据类型的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 XMLTABLE 函数的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	通过按照以下顺序（表、分区、索引、约束、外键和触发器）将转换后的对象应用于目标数据库，改进了迁移流程。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了在源代码中转换注释时意外出现操作项 9996 的问题。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了在转换 FROM 子句别名时意外出现操作项 9997 的问题。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了在转换游标别名时意外出现操作项 9997 的问题。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了转换后的代码为带有 ORDER BY 子句的 SELECT 语句返回不同结果的错误。由于 SQL Server 和 PostgreSQL 对待 NULL 值的方式不同，因此现在转换后的代码包含 NULLS FIRST 或 NULLS LAST 子句，这些子句可确保转换后的代码按与源代码相同的顺序返回结果。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了表函数中的数据类型转换不正确的问题。
MySQL	Amazon RDS for MySQL	解决了转换后的代码中数据库对象名称周围意外出现单引号（' '）的问题。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	在扩展包中添加了新的视图，用于模拟显示有关分区和子分区的信息的 Oracle 系统视图。
Oracle	Aurora PostgreSQL PostgreSQL	更新了扩展包中的两个函数，以便在转换后的代码中添加架构名称作为参数。
Oracle	Aurora PostgreSQL PostgreSQL	修复了在用户界面中刷新应用程序代码后 AWS SCT 未使用正确参数转换 C++ 应用程序的错误。
Oracle	Aurora PostgreSQL PostgreSQL	改进了 `CREATE TYPE` 语句的转换，以避免意外异常。
Oracle	Aurora PostgreSQL PostgreSQL	改进了嵌套表的转换。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	解决了包对象出现的解析错误。
Oracle	Aurora PostgreSQL PostgreSQL	解决了当名称长度超过 60 个字符时，转换后的代码中 AWS SCT 意外修剪了对象名称的问题。
Oracle	Aurora PostgreSQL PostgreSQL	解决了分区表的行级触发器转换不正确的问题。
Oracle 数据仓库	Amazon Redshift	实现了对用于数据迁移的自动表分区的支持。为了加快数据迁移速度，AWS SCT 可以根据ROWID伪列中的值自动对大型表或分区进行分区。有关更多信息，请参阅 使用本机分区。
Teradata	Amazon Redshift	在转换后的 Amazon Redshift 代码中实现了对原生 MERGE 命令的支持。有关 Amazon Redshift MERGE 命令的更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的 MERGE。
Teradata	Amazon Redshift	改进了不使用显式表名的 DELETE 和 UPDATE 语句的转换。
Teradata	Amazon Redshift	解决了 IN 和 NOT IN 语句转换不正确的问题。

B AWS SCT uild 670 的发布说明

来源	目标	新增功能、增强或修复
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了在转换以下代码元素时意外出现操作项 9996 的问题。 • INCLUDE 语句中的 CREATE INDEX 语句 • DECLARE 语句 • DECLARE ... TABLE 语句 • 在 LOOP 语句适用默认值的 DECLARE • DELETE 语句 • ALTER TABLE 语句中的 DROP CONSTRAINT 语句 • EXECUTE AS CALLER 和 REVERT • IIF 语句 • 表达式列表 • MONTH() 函数 • UPDATE 语句 • YEAR() 函数
Azure Synapse Analytics	Amazon Redshift	增加了对作为多服务器评估过程源的 Azure Synapse Analytics 的支持。有关更多信息，请参阅 创建多服务器评估报告。
Hadoop	Amazon EMR	支持在命令行界面（CLI）模式下将 Hadoop 集群迁移到 Amazon EMR。有关更多信息，请参阅 迁移大数据框架。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	修复了源表和列发生的解析器错误。
IBM Db2 for z/OS	Aurora PostgreSQL	实现了 CASE 表达式的转换。

来源	目标	新增功能、增强或修复
	PostgreSQL	
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了将 CURRENT_DATE 引用转换为特殊寄存器。在 Db2 for z/OS 中对特殊寄存器的引用是对当前服务器提供的值的引用。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了 DATE 和 POSSTR 函数的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了日期时间常量的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了以下数据类型列的默认值的转换：DATE、TIME、TIMESTAMP 和 TIMESTAMP WITH TIME ZONE。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了在 SELECT INTO 语句转换过程中意外出现操作项 9996 的问题。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了 DATEDIFF 函数的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了将 ISNULL 函数转换为 NULLIF 的错误。因此，与源代码相比，转换后的代码产生的结果不同。现在， AWS SCT 将ISNULL函数转换为COALESCE。
Netezza	Amazon Redshift	改进了数据提取代理，以解决成功完成的任务却设置为失败状态的问题。
Netezza	Amazon Redshift	添加了在使用数据提取代理开始数据迁移后更改子任务中端点的功能。
Microsoft SQL Server MySQL Oracle PostgreSQL	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	增加了使用 IPv6 地址协议连接到数据库的功能。
Oracle	Amazon RDS for Oracle	实现了计划和管理作业队列中作业的 DBMS_JOB 包的转换。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	在扩展包中添加了新函数，以改进全局嵌套表的转换。这些新函数模拟 Oracle 源代码中的DELETE、EXTEND 和 TRIM 函数。
Oracle	Aurora PostgreSQL PostgreSQL	增加了为嵌入在 Java 应用程序中的 SQL 代码指定转换范围的功能。现在，您可以将源应用程序项目的子集排除在转换范围之外。有关更多信息，请参阅 在 AWS SCT 中转换 Java 应用程序 SQL 代码。
Oracle	Aurora PostgreSQL PostgreSQL	改进了函数索引中联接运算符（||）的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了源代码不包含单个表达式的括号的 IN 条件的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了将 MERGE 语句转换为 PostgreSQL 中的 INSERT ON CONFLICT 语句。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	解决了包出现的解析错误。
Oracle	Aurora PostgreSQL PostgreSQL	解决了在转换包时意外出现操作项 5072 的问题。
Oracle 数据仓库	Amazon Redshift	修复了将转换后的代码应用于目标数据库时 AWS SCT 未应用扩展包的错误。
Oracle 数据仓库	Amazon Redshift	修复了使用扩展包向导时 AWS SCT 未应用某些扩展包文件的错误。
Oracle 数据仓库	Amazon Redshift	解决了在并行运行超过 500 个任务的情况下 AWS SCT 无法处理数据迁移到 AWS Snowball 的问题。
Oracle 数据仓库	Amazon Redshift	解决了用户定义类型的用户定义函数转换不正确的问题。

B AWS SCT uild 669 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	改进了多服务器评估过程，这有助于确定源数据库的最佳目标数据库平台。现在，如果您在输入逗号分隔值 (CSV) 文件中提供数据库凭据，则会 AWS SCT 忽略该 AWS Secrets Manager 密钥。有关更多信息，请参阅 创建多服务器评估报告。

来源	目标	新增功能、增强或修复
全部	全部	解决了使用密钥连接数据库时，多服务器评估报告包含源数据库的 IP 地址的问题。 AWS Secrets Manager
全部	Amazon Redshift	根据操作系统和可用的 RAM，实现了 Java 虚拟机 (JVM) 设置的自动配置。 AWS SCT 使用此 JVM 来运行数据提取代理工作。
全部	Amazon Redshift	解决了数据提取代理无法在 Ubuntu 中启动的问题。
全部	Amazon Redshift	解决了在 Windows 中运行 `StartAgent.bat` 文件后数据提取任务无法启动的问题。
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了生成索引唯一名称选项开启时列名转换不正确的问题。
Greenplum	Amazon Redshift	实现了将 `VOID` 返回过程的函数的转换。
Greenplum	Amazon Redshift	解决了源数据库在数值列中不包含数字 (NaN) 值时数据迁移失败的问题。 AWS SCT 数据提取代理现在将 NaN 值替换为 NULL。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	添加了新的转换设置，用于在转换 `CHAR` 内置函数期间指定 `DATE FORMAT` 和 `TIME FORMAT` 选项。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	添加了操作项 8534，用于转换使用 `WITHOUT RETURN` 子句声明的预定义游标。如果您的光标没有返回结果集，则会在转换后的代码中为光标名称 AWS SCT 赋一个`NULL`值并引发一个操作项。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	编辑了在连接源数据库期间标识 AWS SCT 的 CURRENT CLIENT_APPLNAME 属性。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了新的转换设置，以便在转换 CHAR 内置函数期间指定 DATE FORMAT 和 TIME FORMAT 选项。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了 BEGIN...END 块语句中 LEAVE 语句的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了 XMLPARSE、XMLTABLE 和 XMLNAMESPACES 函数的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 CHAR 内置函数的转换。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了游标的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了 FOR 循环语句转换过程中操作项 9996 意外出现的问题。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了 SELECT 语句中表类型用法的转换。
Microsoft SQL Server	适用于 Aurora PostgreSQL 的 Babelfish	实现了对 Babelfish 功能配置文件新版本 2.2.0 的支持。此文件定义了特定 Babelfish 版本支持和不支持的 SQL 功能。
Netezza	Amazon Redshift	改进了数据提取代理，以解决正在进行的数据复制期间未从目标表中删除一行的问题。
Oracle	Amazon RDS for Oracle	改进了 Oracle 数据企业版功能的转换。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	实现了 GROUPING_ID 函数的转换。
Oracle	Aurora PostgreSQL PostgreSQL	通过在命令行界面（CLI）模式下添加对自定义数据类型映射的支持，改进了 C# 应用程序中的 SQL 代码转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了嵌套表的转换，以避免意外出现操作项 9996。
Oracle	Aurora PostgreSQL PostgreSQL	解决了对象构造函数的调用转换不正确的问题。
Oracle 数据仓库	Amazon Redshift	支持用于数据迁移的现有表分区。为了加快数据迁移速度，请为源表的每个非空分区 AWS SCT 创建子任务。有关更多信息，请参阅 使用本机分区。
Teradata	Amazon Redshift	改进了带有 TIME WITH TIME ZONE AS TIMESTAMP 、TIME WITH TIME ZONE AS CHAR 和 TIMESTAMP AS TIME WITH TIME ZONE 参数的 CAST 函数的转换。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	使用 FORMAT 选项改进了 CAST 函数的转换。
Teradata	Amazon Redshift	解决了 CEIL 函数未转换的问题。
Teradata	Amazon Redshift	解决了带有 DELETE 子句的 MERGE 语句转换不正确的问题。
Teradata	Amazon Redshift	解决了带有日期和格式参数的 TO_CHAR 函数转换不正确的问题。

B AWS SCT uild 668 的发布说明

来源	目标	新增功能、增强或修复
全部	Amazon Redshift	解决了迁移规则中的乘法运算符无法正常运行的问题。这些运算符可以更改 char、varchar、nvarchar 和 string 数据类型的长度。有关更多信息，请参阅 创建迁移规则。
Azure Synapse Analytics	Amazon Redshift	实现了对带 VARCHAR 参数的 CONVERT 函数的支持。
Azure Synapse Analytics	Amazon Redshift	改进了带 NOLOCK 子句的 SELECT 语句的转换。
Azure Synapse Analytics	Amazon Redshift	改进了带别名或带 SET 和 FROM 子句的 UPDATE 语句的转换。
Greenplum	Amazon Redshift	实现了数据迁移自动虚拟分区。 AWS SCT 使用 GP_SEGMENT_ID 系统列创建分区。

来源	目标	新增功能、增强或修复
Greenplum	Amazon Redshift	实现了对 RETURN QUERY 和 RETURN SETOF 子句的支持。
Greenplum	Amazon Redshift	实现了对具有三个参数的 SUBSTRING 函数的支持。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了带 LOCATE 参数的 SUBSTR 函数的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	添加了使用 REFCURSOR 变量数组返回动态结果集的选项。在转换设置中选择此选项时，AWS SCT 将在转换后的代码中添加一个附加 OUT 参数。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了对 FOR 循环语句的支持。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	实现了对 XMLPARSE 函数的支持。为 XMLPARSE 函数中的空格条带化添加了操作项 8541。

来源	目标	新增功能、增强或修复
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了单个 `BEGIN ... END` 块中多个异常处理程序的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了 `INSERT` 和 `DELETE` 触发器的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了嵌套过程调用的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了表类型的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了整数值按位逻辑 `NOT` 运算转换不正确的问题。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了 PostgreSQL 版本 8.0.2 及更低版本中未初始化本地数组的问题。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了带有 `WHEN NOT MATCHED BY SOURCE` 子句的 `MERGE` 语句转换不正确的问题。
MySQL	Aurora MySQL	解决了 AWS SCT 错误确定`rds_superuser_role` 角色授予的用户权限的问题。
Netezza	Amazon Redshift	增强了源元数据加载器，以确保 AWS SCT 正确加载名称为小写的数据库对象。
Oracle	Aurora PostgreSQL PostgreSQL	在扩展包中添加了新函数，以改进本地嵌套表的转换。这些新函数模拟 Oracle 源代码中的 `PRIOR`、`NEXT`、`LIMIT`、`FIRST`、`LAST`、`EXISTS`、`EXTEND`、`TRIM`、`DELET` 和 `SET` 函数。有关更多信息，请参阅 使用扩展包。
Oracle	Aurora PostgreSQL PostgreSQL	增加了为 C# 应用程序指定转换范围的功能。用户现在可以将源应用程序项目的子集排除在转换范围之外。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	实现了对集合中 COUNT 方法的支持。
Oracle	Aurora PostgreSQL PostgreSQL	实现了对嵌套表中变量和构造函数的支持。
Oracle	Aurora PostgreSQL PostgreSQL	实现了对 RATIO_TO_REPORT 和 STANDARD_HASH 函数的支持。
Oracle	Aurora PostgreSQL PostgreSQL	改进了作为 AWS SCT 扩展包一部分的大型对象（LOB）的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了本地集合的转换。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	改进了带有列名不包含表名的 USING 子句的 JOIN 语句的转换。
Oracle	Aurora PostgreSQL PostgreSQL	实现了 EMPTY_BLOB 和 EMPTY_CLOB 函数的转换。
Oracle	Aurora PostgreSQL PostgreSQL	在 C# 应用程序中实现了位置绑定变量的转换。
SAP ASE	Aurora PostgreSQL PostgreSQL	实现了多事件触发器的转换。
SAP ASE	Aurora PostgreSQL PostgreSQL	实现了递归触发器的转换。

来源	目标	新增功能、增强或修复
SAP ASE	Aurora PostgreSQL PostgreSQL	改进了使用 `@@rowcount` 全局变量的触发器的转换。
SAP ASE	Aurora PostgreSQL PostgreSQL	解决了 `UPDATE` 语句 `SET` 子句中的聚合函数转换不正确的问题。
SAP ASE	Aurora PostgreSQL PostgreSQL	解决了在 `UPDATE` 语句转换过程中意外出现操作项 42702 的问题。
SAP ASE	Aurora PostgreSQL PostgreSQL	解决了带有 `CHAR` 参数的 `CONVERT` 函数转换不正确的问题。
Snowflake	Amazon Redshift	增加了对 Snowflake 作为使用数据提取代理进行数据迁移的 AWS SCT 来源的支持。有关更多信息，请参阅 将本地数据仓库中的数据迁移到 Amazon Redshift。
Teradata	Amazon Redshift	改进了带 `TIMESTAMP AS TIME WITH TIMEZONE` 参数的 `CAST` 函数的转换。

AWS SCT 版本 667 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	在命令行界面 (CLI) 模式下实现了对 Informatica 提取、转换和加载 (ETL) 脚本的支持。 AWS SCT 自动将您的 Informatica ETL 脚本重定向到新的目标数据库。此外，还可以 AWS SCT 转换嵌入在 Informatica 对象中的对象名称和 SQL 代码。有关更多信息，请参阅 转换 Informatica ETL 脚本。
全部	Amazon Redshift	将 Amazon Redshift 支持的最低驱动程序版本提高到 2.1.0.9。有关更多信息，请参阅 下载所需的数据库驱动程序。
Azure Synapse Analytics	Amazon Redshift	在扩展包中添加了一个新函数，以改进带有三个日期和时间参数的 `CONVERT` 函数的转换。
Azure Synapse Analytics	Amazon Redshift	改进了 `DATEDIFF` 函数的转换。
Azure Synapse Analytics Microsoft SQL Server 数据仓库	Amazon Redshift	更新了扩展包版本。确保在现有 AWS SCT 项目中应用最新版本的扩展包。有关更多信息，请参阅 使用扩展包。
BigQuery	Amazon Redshift	解决了在命令行界面（CLI）模式下未转换已筛选的对象的问题。
Greenplum	Amazon Redshift	修复了 AWS SCT 未转换存储过程中声明的临时表的错误。
Greenplum	Amazon Redshift	修复了转换后的代码中缺少列编码属性的错误。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	实现了对包含多个 INNER JOIN 子句的自引用表的 UPDATE 语句的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	实现了对 SQL Server 用于 DML 触发器的 inserted 和 deleted 临时表的支持。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了在不同数据库架构中创建的存储过程中用户定义类型的转换。解决了找 AWS SCT 不到数据类型并显示了操作项 9996 的问题。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了转换后的代码中数据库对象名称周围意外出现方括号（[ ]）的问题。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了 @@ROWCOUNT 函数转换不正确的问题。

来源	目标	新增功能、增强或修复
Microsoft SQL Server 数据仓库	Amazon Redshift	实现了对 `geometry` 和 `geography` 数据类型的支持。
Microsoft SQL Server 数据仓库	Amazon Redshift	实现了在转换后的代码中对在数据类型声明中 `MAX` 关键字的支持。
Microsoft SQL Server 数据仓库	Amazon Redshift	改进了 `DATEADD` 函数的转换。
Oracle	Aurora PostgreSQL PostgreSQL	通过添加对 MyBatis框架的支持，改进了 Java 应用程序中的 SQL 代码转换。有关更多信息，请参阅 转换 Java 应用程序中的 SQL 代码。
Oracle	Aurora PostgreSQL PostgreSQL	改进了使用该 MyBatis 框架的 Java 应用程序中的 SQL 代码转换。为语法不支持的 SQL 代码添加了操作项 30411。
Oracle	Aurora PostgreSQL PostgreSQL	通过添加对 `typedef struct` 声明的支持，改进了 Pro*C 应用程序中的 SQL 代码的转换。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	已实现对 `CROSS JOIN` 和 `LEFT JOIN` 语句的支持。
Oracle	Aurora PostgreSQL PostgreSQL	改进了 `MERGE` 语句的转换。解决了转换后的代码中缺少要插入的值的问题。
Teradata	Amazon Redshift	更改了转换后的代码中 AWS SCT 使用的默认列压缩编码设置，以匹配默认 Amazon Redshift 设置。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的压缩编码。
Teradata	Amazon Redshift	解决了使用 `TIME` 数据类型的数学运算转换不正确的问题。
Teradata	Amazon Redshift RSQL	实现了 shell 脚本内部 FastExport 代码的转换。
Teradata BTEQ	Amazon Redshift RSQL	修复了 AWS SCT 未转换`COALESCE`和`%data`语句的错误。
Vertica	Amazon Redshift	改进了用户选择一种优化策略时的转化优化建议。

AWS SCT 版本 666 的发布说明

来源	目标	新增功能、增强或修复
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了 JOIN 语句内的 ON 子句发生的解析错误。
Azure Synapse Analytics	Amazon Redshift	在扩展包中添加了三个新函数，以改进带有日期和时间参数的 CONVERT 函数的转换。
Azure Synapse Analytics	Amazon Redshift	增强了源元数据加载器，以确保 AWS SCT 加载系统数据库架构。
Azure Synapse Analytics	Amazon Redshift	修复了临时表的列出现的解析器错误。
Azure Synapse Analytics	Amazon Redshift	实现了将 BINARY 和 VARBINARY 数据类型转换为 VARBYTE 数据类型。
Azure Synapse Analytics	Amazon Redshift	在转换后的代码中实现了对 TIME 数据类型的支持。
Azure Synapse Analytics	Amazon Redshift	改进了 COLLATE 子句的转换。解决了在使用默认数据库排序规则转换列时意外出现操作项 31141 的问题。
BigQuery	Amazon Redshift	实现了更改输入参数的过程的转换。

来源	目标	新增功能、增强或修复
Greenplum	Amazon Redshift	解决了 AWS SCT 使用与 Greenplum 6.x 数据库不兼容的查询的问题。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	通过将异常处理程序从 Db2 for z/OS 转移到 PostgreSQL，改进了 EXCEPTION 部分的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	改进了 OPEN CURSOR 语句的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	使用 CASE 表达式实现了 IIF 函数的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了当 CREATE PROCEDURE 语句不包含 BEGIN...END 块时带有表值参数的过程转换不正确的问题。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了 SCOPE_IDENTITY 函数转换不正确的问题。

来源	目标	新增功能、增强或修复
Oracle	Amazon RDS for Oracle	修复了使用 Oracle 10g 作为源时 SELECT_CATALOG_ROLE 角色出现的加载器错误。
Oracle	Amazon RDS for Oracle	改进了加载器以支持 Oracle 计划程序作业。
Oracle	Aurora PostgreSQL PostgreSQL	实现了带 USING 子句的 JOIN 语句的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了源代码在 WHERE 子句中包含全局变量的转换后的代码的性能。
Oracle	Aurora PostgreSQL PostgreSQL	通过添加对 MyBatis框架的支持，改进了 Java 应用程序中的 SQL 代码转换。有关更多信息，请参阅 转换 Java 应用程序中的 SQL 代码。
Oracle 数据仓库	Amazon Redshift	实现了 PIVOT 和 UNPIVOT 关系运算符的转换。
Teradata	Amazon Redshift	修复了未转换使用 JSON 对象的源代码的错误。
Teradata	Amazon Redshift	修复了由已删除的用户创建的表未正确加载的错误。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	实现了将 INSTR 函数转换为原生 Amazon Redshift STRPOS 函数。
Teradata	Amazon Redshift	实现了 NVP 和 TRANSLATE 函数的转换。
Teradata	Amazon Redshift	改进了 COALESCE 表达式的转换。
Teradata	Amazon Redshift	改进了 DECLARE CONDITION 语句的转换。
Teradata	Amazon Redshift	使用 SECOND 语法元素改进了 EXTRACT 函数的转换。
Teradata	Amazon Redshift	改进了 LOOP 语句中 SQLSTATE 和 SQLCODE 变量的转换。
Teradata	Amazon Redshift	改进了唯一索引的转换。
Teradata	Amazon Redshift	解决了将小数精度设置为 3 的 CURRENT_TIMESTAMP 语句转换过程中意外出现操作项 9996 的问题。
Teradata	Amazon Redshift	解决了在字符串文字中错误转换反斜杠的问题。
Teradata	Amazon Redshift	解决了转换后的 EXEC 语句在 ADD CONSTRAINT 语句中包含错误的字段名的问题。
Teradata	Amazon Redshift	解决了转换后的 QUALIFY 子查询包含错误的子查询名称的问题。
Teradata	Amazon Redshift	解决了未应用转换后的视图的问题。为转换后的代码中的 NULL 值添加了对特定数据类型的显式强制转换。
Teradata	Amazon Redshift	解决了日期和时间函数转换不正确的问题。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	解决了未转换十六进制字符串文字的问题。

B AWS SCT uild 665 的发布说明

来源	目标	新增功能、增强或修复
Azure Synapse Analytics	Amazon Redshift	实现了带 VARCHAR 参数的 CONCAT 函数的转换。
Azure Synapse Analytics	Amazon Redshift	改进了创建临时表且不包含架构名称的CREATE TABLE语句的转换。AWS SCT 创建dbo架构以将这些临时表存储在目标数据库中。
Azure Synapse Analytics	Amazon Redshift	改进了在临时表上运行的 DROP TABLE 语句的转换。
Azure Synapse Analytics	Amazon Redshift	改进了带有 BEGIN...END 块的 OBJECT_ID 语句的转换。
Azure Synapse Analytics	Amazon Redshift	解决了 AWS SCT 无法转换带有区块注释的存储过程的错误。
BigQuery	Amazon Redshift	实现了将 BigQuery 数据仓库转换为亚马逊 Redshift。有关更多信息，请参阅 使用 BigQuery 作为 AWS SCT 的源。
Microsoft SQL Server	Aurora PostgreSQL	改进了处理多个事件并在 SQL Server 中与 inserted 和 deleted 系统表配合使用的触发器的转换。

来源	目标	新增功能、增强或修复
	PostgreSQL	
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了 SQL Server 中 `inserted` 和 `deleted` 系统表发生的解析器错误。
Microsoft SQL Server	适用于 Aurora PostgreSQL 的 Babelfish	实现了对 Babelfish 功能配置文件新版本 2.1.0 的支持。此文件定义了特定 Babelfish 版本支持和不支持的 SQL 功能。
Oracle	Aurora MySQL MariaDB MySQL	解决了 `varchar2` 数据类型转换不正确的问题。
Oracle	Aurora MySQL Aurora PostgreSQL MariaDB MySQL PostgreSQL	对于 12c 及更高版本的 Oracle 数据库，AWS SCT 支持以下扩展数据类型： • `VARCHAR2` • `NVARCHAR2` • `RAW` AWS SCT 将这些数据类型支持的最大列长度从 8,000 字节增加到 32,767 字节。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	解决了 Oracle 事件处理包发生的解析错误。
Teradata	Amazon Redshift	为单个 SELECT 语句中的多个 RESET WHEN 子句添加了操作项 13214。
Teradata	Amazon Redshift	为位于异常处理块之外的 SQLSTATE 变量添加了操作项。
Teradata	Amazon Redshift	实现了将 ACTIVITY_COUNT 变量转换为 ROW_COUNT 。
Teradata	Amazon Redshift	实现了内置几何 ST_TRANSFORM 函数的转换。
Teradata	Amazon Redshift	改进了不带 WHERE 子句的视图中删除语句的转换。
Teradata	Amazon Redshift	改进了表达式中 CAST 运算符的转换。
Teradata	Amazon Redshift	改进了 GROUP BY 子句的转换。
Teradata	Amazon Redshift	改进了 INSTR 和 REGEXP_INSTR 内置函数的转换。
Teradata	Amazon Redshift	解决了横向列别名引用转换不正确的问题。
Teradata	Amazon Redshift	解决了 QUALIFY 子查询中列名转换不正确的问题。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	实现了使用 ERRORCODE 状态值关键字的 .QUIT 命令转换。
Teradata BTEQ	Amazon Redshift RSQL	解决了在 CREATE 语句转换过程中意外出现操作项 9996 的问题。
Teradata BTEQ	Amazon Redshift RSQL	解决了在 END 语句转换过程中意外出现操作项 9998 的问题。

AWS SCT 版本 664 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	增加了对 Amazon Redshift Serverless 作为 AWS SCT中数据库迁移项目的源和目标的支持。要连接到 Amazon Redshift Serverless，请确保您使用 Amazon Redshift JDBC 驱动程序 2.1.0.9 或更高版本。
全部	全部	改进了转换设置窗口的用户界面。 AWS SCT 现在仅显示带有已创建映射规则的数据库转换对的设置。有关更多信息，请参阅 创建映射规则。
全部	全部	更新了评估报告，以删除了有关操作项行和位置的重复信息。
全部	Amazon Redshift	在数据提取任务中实现了自动内存平衡。
全部	Amazon Redshift	解决了数据提取代理无法连接到 AWS Snowball 设备的错误。
Azure SQL 数据库 IBM Db2 for z/OS	Aurora MySQL	支持 SUSE Linux 15.3 作为运行数据提取代理的平台。

来源	目标	新增功能、增强或修复
IBM Db2 LUW Microsoft SQL Server MySQL Oracle PostgreSQL SAP ASE	Aurora PostgreSQL MariaDB MySQL PostgreSQL	
Azure Synapse Analytics	Amazon Redshift	改进了 DATEADD 函数的转换。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	添加了更改迁移规则中列排序规则的功能。
Microsoft SSIS	AWS Glue AWS Glue Studio	解决了用户选择源脚本时发生的意外错误。
Oracle	Aurora MySQL MariaDB MySQL	实现了将存储函数的用法转换为生成的列表达式。 AWS SCT 创建触发器来模拟此行为，因为 MySQL 不支持使用存储的函数作为生成的列表达式。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	作为 AWS SCT 扩展包的一部分，实现了对UTL_MATCH 包中函数的转换。
Oracle	Aurora PostgreSQL PostgreSQL	实现了待 NULL 参数的 REGEXP_LIKE 函数的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了 SYS_EXTRACT_UTC 函数的转换。
Oracle	Aurora PostgreSQL PostgreSQL	通过支持 Wcscats、Wcscpys 和 Wcsncats 函数，改进了 C++ 应用程序中的 SQL 代码转换。有关更多信息，请参阅 使用 AWS SCT 转换 C++ 应用程序中的 SQL 代码。
Oracle 数据仓库 Snowflake	Amazon Redshift	解决了转换后的语句不包括将值显式转换为列数据类型的问题。使用来自其他表的查询结果的语句中出现了此问题。
Teradata	Amazon Redshift	增加了在迁移规则中更改 case sensitive 和 case insensitive 之间的列排序规则的功能。有关更多信息，请参阅 创建迁移规则。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	修复了 CREATE TABLE AS 语句中出现的解析器错误。
Teradata	Amazon Redshift	修复了未转换带有 COALESCE 表达式的内置 P_INTERSECT 函数的错误。
Teradata	Amazon Redshift	实现了将名为 OID 的列转换为 _OID，以避免在 Amazon Redshift 中使用保留关键词。
Teradata	Amazon Redshift	实现了函数、过程、视图和宏 RENAME 语句的转换。
Teradata	Amazon Redshift	在 Amazon Redshift 中实现了将 STROKE 函数转换为 SPLIT_PART 函数。
Teradata	Amazon Redshift	改进了 INSTR 和 REGEXP_INSTR 系统函数的转换。
Teradata	Amazon Redshift	改进了 TIME 数据类型的转换。
Teradata	Amazon Redshift	通过实现一级和二级唯一索引的转化，改进了 SET 和 MULTISET 表的模拟。
Teradata	Amazon Redshift	解决了 CHARACTER 函数发生的解析错误。
Teradata BTEQ	Amazon Redshift RSQL	解决了用户从项目中删除 Teradata Basic Teradata 查询 (BTEQ) 脚本时发生的错误。 AWS SCT

AWS SCT 版本 663 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	添加了在迁移规则中使用乘法运算符更改 `char`、`varchar`、`nvarchar` 和 `string` 数据类型长度的功能。有关更多信息，请参阅 创建迁移规则。
全部	全部	在多服务器评估报告中支持三个新列，并更新了输入文件的格式。请务必使用最新版本的 AWS SCT更新输入文件模板。有关更多信息，请参阅 创建数据库迁移多服务器评估报告。
Azure Synapse Analytics	Amazon Redshift	改进了 `OBJECT_ID` 语句的转换。
Microsoft SQL Server	适用于 Aurora PostgreSQL 的 Babelfish	增加了对适用于 Aurora PostgreSQL 的 Babelfish 1.2.0 作为数据库迁移评估报告的目标平台的支持。有关更多信息，请参阅《Amazon Aurora 用户指南》中的按版本划分的 Babelfish 支持的功能。
Microsoft SQL Server 数据仓库	Amazon Redshift	增加了对 `AT TIME ZONE` 子句的支持。
Microsoft SQL Server 数据仓库	Amazon Redshift	解决了 `BEGIN/END` 块外的语句转换不正确的问题。
Netezza	Amazon Redshift	改进了 `TIME` 数据类型的转换，实现了相关内置函数、表达式和文字的转换。
Oracle	Aurora PostgreSQL	修复了使用 Oracle 10g 作为源时发生的加载器错误。

来源	目标	新增功能、增强或修复
	PostgreSQL	
Oracle	Aurora PostgreSQL PostgreSQL	改进了 OFFSET 和 FETCH 子句的转换。
Oracle	Aurora PostgreSQL PostgreSQL	解决了 OUT 参数为默认值的过程转换不正确的问题。
Oracle 数据仓库	Amazon Redshift	改进了 Oracle 函数到 Amazon Redshift 用户定义函数的转换。
Snowflake	Amazon Redshift	改进了 WITH 子句的转换。
Teradata	Amazon Redshift	为 CHAR 数据类型不支持的多字节字符添加了新的操作项 13209。
Teradata	Amazon Redshift	修复了表格未完全加载的加载器错误。
Teradata	Amazon Redshift	修复了 JOIN 条件下的内置 P_INTERSECT 函数未转换的转换程序错误。
Teradata	Amazon Redshift	修复了在名称中包含特殊字符的表上运行 SELECT 语句时视图名称在错误的情况下转换的问题。
Teradata	Amazon Redshift	改进了具有 PERIOD(DATE) 数据类型 UNTIL_CHANGED 值的 INSERT 语句的转换。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	使用 Amazon Redshift 中的 TO_CHAR 函数改进了内置 FORMAT 函数的转换。
Teradata	Amazon Redshift	改进了内置 RANK 函数的转换，以确保转换后的代码按与源代码相同的顺序返回 NULL 值。
Teradata	Amazon Redshift	改进了唯一约束（例如一级或二级唯一索引）的转换。

AWS SCT 版本 662 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	增加了在创建多服务器评估报告时自动为每个源数据库创建 AWS SCT 项目的功能。启用此选项后，AWS SCT 可以向这些项目添加映射规则并保存转化统计信息以供离线使用。有关更多信息，请参阅 创建数据库迁移多服务器评估报告。
全部	全部	在创建多服务器评估报告时，支持在数据库和架构名称中使用百分比（%）作为通配符。
全部	Aurora MySQL Aurora PostgreSQL	已将所有 AWS Lambda 函数的运行时更新到 Python 版本 3.9。
全部	Amazon Redshift	已升级所有要使用的数据提取代理 AWS SDK for Java 2.x。
Azure SQL 数据库	Aurora PostgreSQL	改进了带 NOT EXISTS 子句的 DELETE 语句的转换。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	PostgreSQL	
Azure Synapse Analytics	Amazon Redshift	解决了源数据库连接失败的错误。
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	解决了转换后的触发器代码中两次提及对象别名的错误。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了当数据库对象名称区分大小写选项处于开启状态时名称为大小写混合的对象的转换。
Microsoft SQL Server 数据仓库 Teradata	Amazon Redshift	实现了 `PIVOT` 和 `UNPIVOT` 关系运算符的转换。
Netezza	Amazon Redshift	实现了 `TIME` 数据类型的转换。

来源	目标	新增功能、增强或修复
Oracle	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	实现了 UTL_TCP.CRLF 包常量的转换。
Oracle	Aurora PostgreSQL PostgreSQL	修复了扩展包问题，即在转换过程中可变长度列的数据类型长度未保持不变。
Oracle	Aurora PostgreSQL PostgreSQL	在 C++ 应用程序中实现了 SQL 代码转换。有关更多信息，请参阅 使用 AWS SCT 转换 C++ 应用程序中的 SQL 代码。
Oracle	Aurora PostgreSQL PostgreSQL	支持全局变量和关联数组转换时区分大小写命名。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	改进了扩展包中 TO_CHAR、TO_DATE 和 TO_NUMBER 函数的转换。
Oracle	Aurora PostgreSQL PostgreSQL	改进了 TABLE() 运算符的转换。
Oracle 数据仓库	Amazon Redshift	增加了对主键和其他约束转换的支持。
Oracle 数据仓库	Amazon Redshift	修复了在条件语句转换过程中操作项 12054 不显示的问题。
SAP ASE	Aurora PostgreSQL PostgreSQL	解决了在转换包含用户定义类型列的表期间在目标树中创建了名称为空的对象时出现的错误。
SAP ASE	Aurora PostgreSQL PostgreSQL	修复了脚本、例程等存储对象的加载器错误。
Snowflake	Amazon Redshift	修复了操作项目 22152 在需要时不显示并将转换结果 AWS SCT 显示为评论的问题。

来源	目标	新增功能、增强或修复
Snowflake	Amazon Redshift	改进了日期和时间函数的转换；实现了对时区的支持。
Snowflake	Amazon Redshift	解决了带 WITH 子句的非递归公用表表达式（CTE）被转换为递归公用表表达式的问题。
Teradata	Amazon Redshift	改进了带条件中表链接的 UPDATE 语句的转换。
Teradata	Amazon Redshift	改进了 RENAME TABLE 语句的转换。
Teradata	Amazon Redshift	解决了评估报告中逗号分隔值（CSV）文件中显示空列的问题。
Teradata	Amazon Redshift RSQL	修复了转换后的 Basic Teradata Query（BTEQ）宏末尾缺少分号的错误。
Teradata	Amazon Redshift RSQL	改进了 CASE 语句中多个数据类型值的转换。
Teradata	Amazon Redshift RSQL	改进了带 ESCAPE 字符的 LIKE ANY 子句的转换。
Teradata	Amazon Redshift RSQL	改进了 INSERT 语句中 CAST 函数的转换。
Teradata	Amazon Redshift RSQL	改进了时区的转换，实现了时区区域映射。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift RSQL	解决了在 Shell 脚本与 BTEQ 脚本转换过程中意外出现操作项 9998 的问题。
Teradata	Amazon Redshift RSQL AWS Glue	限制替代变量值不超过 500 个字符。
Vertica	Amazon Redshift	实现了将 BINARY、VARBINARY 、LONG BINARY、BYTEA 和 RAW 数据类型转换为 VARBYTE 数据类型。
Vertica	Amazon Redshift	改进了内置函数和文字的转换。

B AWS SCT uild 661 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	添加了用于在映射视图中搜索映射规则的过滤器。应用筛选器时，AWS SCT 会显示与服务器映射列表中的筛选条件相匹配的规则。有关更多信息，请参阅 管理映射规则。
全部	全部	将 Apache Log4j 升级到版本 2.17.1 。
全部	Amazon Redshift	增加了对使用 COPY 命令中的 ENCRYPTED 子句将数据迁移到 Amazon Redshift 的支持。
全部	Amazon Redshift	增强了数据提取代理的 REST API。更新后的 REST API 增加了对加密密钥、加密类型等新属性的支持。
全部	Amazon Redshift	在数据提取代理中实现了角色代入。此更新改善了子任务的分配，并允许 AWS SCT 将任务分配给指定角色的自由代理。

来源	目标	新增功能、增强或修复
全部	Amazon Redshift	在将扩展包应用于 Amazon Redshift 之前，检查是否已安装所有必需的组件。
Azure Synapse Analytics Microsoft SQL Server 数据仓库	Amazon Redshift	改进了用于错误处理的 `ERROR_LINE` 、`ERROR_MESSAGE` 、`ERROR_NUMBER` 、`ERROR_PROCEDURE` 、`ERROR_SEVERITY` 和 `ERROR_STATE` 系统函数的转换。
IBM Db2 for z/OS	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	增加了对 IBM Db2 for z/OS 版本 12 的作为 AWS SCT中数据库迁移项目的源的支持。有关更多信息，请参阅 将 IBM Db2 for z/OS 用作源。
IBM Db2 LUW	全部	增强了源元数据加载器，以确保 AWS SCT 加载与列名重复的例行参数。
Microsoft Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了使用 `SET NOCOUNT ON` Set 语句的过程的转换程序错误。

来源	目标	新增功能、增强或修复
Microsoft Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了当输入值为用户定义类型的变量时 CONCAT 函数的转换。
Microsoft Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了 DATEPART 函数转换不正确的问题。
Microsoft SQL Server	适用于 Aurora PostgreSQL 的 Babelfish	实现了对 Babelfish 功能配置文件新版本的支持。此文件定义了特定 Babelfish 版本支持和不支持的 SQL 功能。
Microsoft SQL Server 数据仓库	Amazon Redshift	解决了带有 EXECUTE 语句的过程转换不正确的问题。
Microsoft SSIS	AWS Glue	改进了作业配置向导的用户界面。 AWS SCT 现在，在连接配置部分中仅显示可用的连接。
Microsoft SSIS	AWS Glue	解决了转换规则未应用于包任务和变量规则的问题。
Microsoft SSIS	AWS Glue AWS Glue Studio	为不支持的组件添加了新的操作项 25042。

来源	目标	新增功能、增强或修复
Microsoft SSIS	AWS Glue Studio	实现了将微软 SQL Server 集成服务 (SSIS) 提取、转换和加载 (ETL) 包转换为。 AWS Glue Studio有关更多信息，请参阅 将 SSIS 转换为 AWS Glue Studio。
Oracle	MariaDB	修复了 MINUS 运算符转换的问题。
Oracle	MariaDB	改进了 MariaDB 中的 sql_mode 系统变量到 Oracle 时 ROWNUM、SYS_GUID、TO_CHAR 和 ADD_MONTHS 函数的转换。
Oracle	PostgreSQL	添加了避免在通用应用程序转换项目中将绑定变量类型转换为 SQL 类型的选项。
Oracle	PostgreSQL	添加了避免在通用应用程序转换项目中将架构名称添加到转换后的对象的名称中的选项。
Oracle	PostgreSQL	为应用程序 SQL 代码转换添加了对 ?x 绑定变量格式的支持。
Oracle 数据仓库	Amazon Redshift	实现了将 RAW 数据类型转换为 VARBYTE 数据类型。
Teradata	Amazon Redshift	添加了在转换后的代码中模拟 SET 表的选项。对于此仿真，请使用 AWS SCT 支撑MIN和MAX条件。
Teradata	Amazon Redshift	改进了具有不同数据类型参数的联接操作的转换。此更新 AWS SCT 允许在转换此类操作期间应用转换规则。
Teradata	Amazon Redshift	解决了 GROUP BY 子句转换不正确的问题。
Teradata	Amazon Redshift	解决了 QUALIFY 子句转换不正确的问题。
Teradata	Amazon Redshift	解决了 FastExport 脚本导入过程中发生的意外错误。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift RSQL	实现了在 Teradata BTEQ 和 Shell 脚本中编辑变量值的功能。
Teradata	Amazon Redshift RSQL	解决了转换后的 Teradata FastLoad 会话缺少清单脚本的问题。
Teradata	Amazon Redshift RSQL	解决了转换后的FastLoad 脚本的统一资源定位器 (URL) 中缺少清单文件扩展名的问题。
Teradata BTEQ	Amazon Redshift RSQL	修复了带有替代变量的脚本加载器错误。
Teradata BTEQ	Amazon Redshift RSQL	修复了操作项 27022 在需要时不会出现的问题。

B AWS SCT uild 660 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	在多服务器评估报告中增加了对 AWS Secrets Manager 和安全套接字层 (SSL) 的支持。有关更多信息，请参阅 创建数据库迁移多服务器评估报告。
全部	全部	改进了转换后的对象的统计数据收集。
全部	PostgreSQL	支持将 PostgreSQL 主要版本 14 和 MariaDB 10.6 作为迁移目标。

来源	目标	新增功能、增强或修复
Azure Synapse Analytics	Amazon Redshift	改进了转换后对象的名称的转换逻辑。
Microsoft Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL	改进了 XML 数据类型的转换。
Microsoft Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了 NOT LIKE 子句转换不正确的问题。
Microsoft Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了带有包含 OUTPUT 子句的 INSERT、DELETE 和 UPDATE 语句的过程的转换程序错误。
Microsoft Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了带有 RETURN @@ROWCOUNT 语句的过程的转换程序错误。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	全部	改进了使用链接服务器的过程的转换。
Microsoft SQL Server	全部	在多服务器评估报告中增加了对 Microsoft Windows 身份验证的支持。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了表值构造函数的转换程序错误。
Microsoft SQL Server 数据仓库	亚马逊 Redshift 和 AWS Glue	改进了提取、转换和加载（ETL）脚本的转换，以包括转换后脚本的正确路径。
Microsoft SQL Server 数据仓库	Amazon Redshift	解决了为虚拟和真实目标数据库平台生成不同的转换后的脚本的问题。
Oracle	PostgreSQL Aurora PostgreSQL	增加了对实体化视图索引转换的支持。

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL Aurora PostgreSQL	修复了在使用 NOVALIDATE 选项转换 PRIMARY KEY 和 UNIQUE 约束时操作项 5982 不显示的问题。
Oracle 数据仓库	Amazon Redshift	解决了转换后的架构中显示其他类别的问题。
Teradata	Amazon Redshift	修复了将未解析的列转换为 CAST 函数的参数时不显示操作项 13185 的问题。
Teradata	Amazon Redshift	改进了 DELETE 和 DELETE ALL 语句的转换，以便在转换后的代码中使用 TRUNCATE 命令。
Teradata	Amazon Redshift	改进了 SET 表的转换。
Teradata	Amazon Redshift	改进了 NORMALIZE 条件的转换。
Teradata	Amazon Redshift	更新了评估报告，将数据库架构转换统计信息从数据库存储对象列表中删除。
Teradata	Amazon Redshift	改进了不带 FROM 子句的 UPDATE 语句的转换。
Teradata	Amazon Redshift	在转换后的代码中实现了对 VARBYTE 数据类型的支持。
Teradata BTEQ	AWS Glue	解决了在上下文菜单中禁用转换为 AWS Glue 选项的问题。
Teradata BTEQ	Amazon Redshift RSQL	解决了转换后的代码中缺少数据类型的问题。

来源	目标	新增功能、增强或修复
Teradata BTEQ	Amazon Redshift RSQL	解决了在转换后的代码中错误引用替代变量的问题。
Teradata BTEQ	Amazon Redshift RSQL	修复了将替换变量与FastLoad 脚本中的值进行转换时出现的问题。
Vertica	Amazon Redshift	在转换后的代码中实现了对 TIME 数据类型的支持。
Vertica	Amazon Redshift	改进了 SELECT DISTINCT 和 ORDER BY 表达式的转换。
Vertica	Amazon Redshift	增加了对约束转换的支持。
Vertica	Amazon Redshift	解决了评估报告未保存为逗号分隔值（CSV）文件的问题。

B AWS SCT uild 659 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	改进了新建项目向导，该向导可以为多个源数据库生成合并评估报告。
全部	全部	修复了在包含多个源数据库和目标数据库的项目中未创建扩展包的问题。
全部	全部	改进了嵌入在应用程序源代码中的 SQL 代码的转换。
全部	全部	增加了在 AWS SCT 命令行界面中从不同文件夹运行脚本的功能。
全部	Amazon Redshift	改进了用户在使用 Amazon Redshift 虚拟目标数据库平台的迁移项目中选择运行优化时提供的警告消息。

来源	目标	新增功能、增强或修复
全部	Aurora PostgreSQL	在 Aurora PostgreSQL 兼容版本上支持将 PostgreSQL 主要版本 13 作为迁移目标。
全部	Amazon RDS for MySQL	实现了默认情况下不区分大小写的代码转换。
Azure Synapse Analytics	Amazon Redshift	解决了命令行界面中源数据库连接失败的错误。
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	改进了包括带有联接条件的 UPDATE 语句的过程的转换。
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	改进了触发器、存储过程和包含等号后面的值的函数的转换。
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	修复了带 DELETE 语句和 OR 运算符的过程的转换程序错误。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	改进了 OUTPUT 子句的转换。
Microsoft SQL Server 数据仓库	亚马逊 Redshift 和 AWS Glue	改进了 NUMERIC 数据类型的转换。
Microsoft SQL Server 数据仓库	Amazon Redshift	改进了表别名与原始表同名的视图的转换。
Microsoft SSIS	AWS Glue	修复了在 “配置 AWS Glue 连接” 窗口中未显示连接凭据的问题。
Netezza	Amazon Redshift	增加了每天重复执行变更数据捕获（CDC）数据迁移任务的功能。
Netezza	Amazon Redshift	修复了取消注册数据提取代理后任务选项卡变为非活动状态的问题。
Netezza	Amazon Redshift	修复了用户界面中未显示数据迁移代理注册确认信息的问题。
Netezza	Amazon Redshift	修复了源数据库连接失败并出现加载器错误的问题。
Netezza	Amazon Redshift	解决了打开已保存的项目后数据迁移代理无法运行的错误。

来源	目标	新增功能、增强或修复
Oracle	Amazon RDS for Oracle	支持 Oracle 统一审计。
Oracle	PostgreSQL Aurora PostgreSQL	在 C# 应用程序中实现了 SQL 代码转换。有关更多信息，请参阅 转换 C# 应用程序中的 SQL 代码。
Oracle	PostgreSQL Aurora PostgreSQL	为区分大小写的对象名称实现了新的转换逻辑，以提高代码转换更改的可见性。 AWS SCT 将大写的对象名称转换为小写。反之亦然；AWS SCT 将小写的对象名称转换为大写。其他对象名称和保留字未经更改即可进行转换。
Oracle	PostgreSQL Aurora PostgreSQL	改进了无 `NOT NULL` 约束的哈希分区的转换。
Oracle	Aurora PostgreSQL	添加了对带有 `ENABLE NOVALIDATE` 子句的 Oracle `CHECK`、`FOREIGN KEY` 和 `NOT NULL` 约束的支持。
Oracle 数据仓库	Amazon Redshift	修复了迁移不正确的浮点数值的问题。
Oracle 数据仓库	亚马逊 Redshift 和 AWS Glue	解决了逗号分隔值（CSV）文件中数据库迁移评估报告中的空列的问题。

来源	目标	新增功能、增强或修复
SAP ASE	PostgreSQL Aurora PostgreSQL	修复了转换意外中断的问题。
Snowflake	Amazon Redshift	改进了 VARIANT 数据类型的转换。
Teradata	Amazon Redshift	改进了 COLLECT STATISTICS 语句的转换。
Teradata	Amazon Redshift	修复了转换带 PERIOD 列的嵌套视图时操作项 9998 不显示的问题。
Teradata	亚马逊 Redshift 和 AWS Glue	修复了打开已保存的项目后，虚拟 AWS Glue 目标平台未显示在界面中的问题。
Teradata BTEQ	AWS Glue	修复了打开已保存的项目后不支持转换为虚拟 AWS Glue 目标平台的问题。
Teradata BTEQ	Amazon Redshift RSQL	改进了转换后的代码的语法高亮显示。
Teradata BTEQ	Amazon Redshift RSQL	实现了上传后检查参数值。变量选项卡上会突出显示不支持的值。
Vertica	Amazon Redshift	实现了聚合函数的转换。
Vertica	Amazon Redshift	实现了将投影转换为实体化视图的功能，并改进了显示投影源代码的用户界面。

B AWS SCT uild 658 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	提供与的集成 AWS Secrets Manager。现在，您可以使用存储在 Secrets Manager 中的数据库连接凭证。
全部	全部	在 AWS SCT 命令行界面中添加了对 YAML 格式脚本的支持。
全部	Amazon Redshift	在数据提取代理中实现了对 Amazon S3 接口端点 (VPCE) 的支持。
全部	Amazon Redshift	除了已经支持的亚马逊 Redshift 和组合之外，还增加了对亚马逊 Red AWS Glue shift 虚拟目标数据库平台的支持。
Greenplum	Amazon Redshift	修复了另存为 SQL 选项未将转换后的 SQL 代码保存到文件的问题。
IBM Db2 LUW Microsoft Azure SQL 数据库 Microsoft SQL Server Oracle SAP ASE	Aurora MySQL	改进了转换以支持与 MySQL 8.0 兼容的 Amazon Aurora MySQL 兼容版本的新功能。
Microsoft SQL Server	Aurora MySQL Aurora PostgreSQL	修复了操作项 810 在需要时不会出现的问题。

来源	目标	新增功能、增强或修复
	MySQL PostgreSQL	
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了带 UPDATE、DELETE 和 INSERT 语句的过程的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了操作项 7810 在需要时不会出现的问题。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了嵌套在 IF...ELSE 语句中的 EXEC 语句的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了索引视图的转换。
Netezza	Amazon Redshift	通过跟踪更改数据捕获（CDC）操作中满负荷期间的实时事务来改进数据迁移代理。现在，如果 CDC 会话计划在特定时间开始，则可以停止数据迁移任务。此外，在使用 CDC 停止任务后，您还可以在控制台中看到错误日志记录级别。

来源	目标	新增功能、增强或修复
Oracle	全部	增强了表格加载器，以确保 AWS SCT 加载带有共享选项的对象。
Oracle	Aurora PostgreSQL PostgreSQL	改进了 SYSDATE 函数的转换，并增加了在转换设置中更改时区的功能。
Oracle	Aurora PostgreSQL PostgreSQL	解决了未转换动态语句的问题。
Oracle	Aurora PostgreSQL PostgreSQL	修复了转换后的代码不包含系统生成的名称的问题。
Oracle Oracle 数据仓库	Aurora PostgreSQL PostgreSQL	改进了嵌套在触发器中的 SELECT 语句的转换。
Oracle 数据仓库	Amazon Redshift	改进了扩展包中 TO_DATE、TO_TIMESTAMP 和 TO_TIMESTAMP_TZ 函数的转换。
Snowflake	Amazon Redshift	添加了一个选项，用于将转换后的 SQL 代码保存到每个对象或每个语句的不同文件中。
Teradata	Amazon Redshift	改进了 CONCAT 函数的转换。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	改进了嵌套在 WHERE 子句中的 SELECT 语句的转换。
Teradata	Amazon Redshift	解决了用户删除并重新创建表后 SET 和 MULTISET 表转换不正确的问题。
Teradata	Amazon Redshift	改进了包含 WITH 子句的过程的转换。
Teradata	Amazon Redshift	改进了 DATE 数据类型的转换。
Teradata	Amazon Redshift RSQL	解决了FastExport 脚本转换过程中发生意外转换器错误的问题。
Teradata BTEQ	Amazon Redshift RSQL	添加了对将联接索引转换为实体化视图的支持。
Teradata BTEQ	Amazon Redshift RSQL	增加了对包含多行的 TITLE 定义的转换的支持。
Teradata BTEQ	Amazon Redshift RSQL	解决了未转换地理空间数据类型大小的问题。
Teradata BTEQ	Amazon Redshift RSQL	修复了参数名称转换为小写字符的问题。
Teradata BTEQ	Amazon Redshift RSQL	修复了嵌套在 MACRO 语句中的存储过程未转换的问题。

来源	目标	新增功能、增强或修复
Vertica	Amazon Redshift	改进了 ALL 运算符的转换。
Vertica	Amazon Redshift	解决了未应用转化设置中的 Use Union all view? 选项的问题。
Vertica	Amazon Redshift	改进了 TIME 和 TIME WITH TIMEZONE 数据类型的转换。
Vertica	Amazon Redshift	解决了加载弹性表时的问题。

已解决的问题：

- 常规改进。

B AWS SCT uild 657 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	已将 Apache Log4j 升级到版本 2.17，以解决安全漏洞问题。
全部	Amazon Redshift	改进了架构优化项目，其中密钥管理统计数据未保存在 AWS SCT 项目中。
Amazon Redshift	Amazon Redshift	修复了服务器信息更新问题。
Apache Cassandra	Amazon DynamoDB	修复了使用 AWS SCT 命令行界面时映射规则的问题。
Apache Cassandra	Amazon DynamoDB	解决了由于证书中的标题更新而未创建迁移任务的问题。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了一个问题，以便在使用动态 SQL 转换 Microsoft SQL Server 过程期间操作项 7672 不会出现。
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了表值函数的转换。
Azure SQL 数据库 Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	解决了使用默认返回值的存储过程中的 OUT 参数未转换为 INOUT 参数的问题。
Greenplum	Amazon Redshift	通过从 QueryLog 表中查找最常用的表和列，改进了优化策略。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了以下内容的转换问题： • 字符串连接赋值运算符（+=） • SCOPE_IDENTITY 函数 • varchar(max) 数据类型

来源	目标	新增功能、增强或修复
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	改进了使用不支持的函数的视图的转换。
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	修复了另一个函数的参数的不支持的函数转换不正确的问题。
Microsoft SQL Server	适用于 Aurora PostgreSQL 的 Babelfish	改进了过渡表引用的转换。
Microsoft SQL Server 数据仓库	Amazon Redshift	在源数据库元数据树中添加了聚合函数类别。
Microsoft SQL Server 数据仓库	Amazon Redshift	改进了 `TIME` 数据类型的转换。

来源	目标	新增功能、增强或修复
Azure Synapse Analytics Greenplum Netezza Microsoft SQL Server 数据仓库 Snowflake Teradata	Amazon Redshift	修复了使用虚拟目标数据库平台时 DROP 和 CREATE 脚本未保存的问题。
Microsoft SQL Server Integration Services	AWS Glue	解决了源对象的脚本未显示在用户界面中的问题。
Netezza	Amazon Redshift	通过为排序规则选择事实数据表和适当的维度，改进了优化策略。
Oracle	Aurora PostgreSQL PostgreSQL	解决了不正确转换使用序列号的 Oracle 触发器的问题。

来源	目标	新增功能、增强或修复
Oracle	Aurora PostgreSQL PostgreSQL	改进了带有公共数据库链接的视图的转换。
Oracle 数据仓库	Amazon Redshift	通过分析索引列的基数，改进了优化策略。
Oracle 数据仓库	Amazon Redshift	修复了带有字符串连接的自定义用户定义标量函数被转换不正确的问题。
Snowflake	Amazon Redshift	修复了用户界面中未显示另存为 SQL 选项的问题。
Teradata	Amazon Redshift	修复了统计数据收集失败并出现 `LOADER ERROR` 异常的问题。
Teradata	Amazon Redshift	修复了用户界面中未显示创建报告选项的问题。
Teradata	Amazon Redshift	改进了 `CAST` 函数的转换。
Teradata	Amazon Redshift	修复了 `ST_Line_Interpolate_Point` 转换失败的问题。
Teradata	Amazon Redshift	从 Python 库路径中删除了一个意外值。
Teradata	Amazon Redshift RSQL	修复了在转换多个 FastLoad脚本时出现的解析器错误。

来源	目标	新增功能、增强或修复
Teradata BTEQ	Amazon Redshift RSQL	改进了 DATABASE 命令和几何数据类型的转换。
Teradata BTEQ	AWS Glue	修复了用户界面中源脚本和目标脚本同步不正确的问题。

已解决的问题：

- 常规改进。

B AWS SCT uild 656 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	在一个项目中增加了对多个源数据库和目标数据库的支持。用户现在可以创建映射规则，以匹配同一个项目中的不同数据库架构和目标平台。
全部	全部	增加了对虚拟目标数据库平台的支持。现在，用户无需连接到目标数据库即可查看其源数据库架构的 AWS SCT 转换情况。
全部	全部	用户界面改进： • 在源元数据树和目标元数据树中添加了连接服务器和断开与服务器的连接选项。 • 添加了从 AWS SCT 项目中移除数据库服务器的选项。
Cassandra	Amazon DynamoDB	解决了 CASSANDRA_HOME 变量在 cassandra.yaml 或 conf 文件夹后面没有斜杠（/）的搜索问题。

来源	目标	新增功能、增强或修复
Cassandra	Amazon DynamoDB	增加了对 Amazon Linux 2 的亚马逊机器映像（AMI）的支持。
Cassandra	Amazon DynamoDB	改进了为 Cassandra 提供的密钥不正确时出现的错误消息。
Cassandra	Amazon DynamoDB	通过根据目标数据库的版本更改 `cassandra-env.yaml` 文件中的属性来改进转换。
Cassandra	Amazon DynamoDB	将目标 Cassandra 数据中心的 Java 版本提高到 1.8.0。
Greenplum	Amazon Redshift	改进了项目设置中的优化策略。
Greenplum	Amazon Redshift	解决了数据迁移问题，其中对象未应用于数据库并出现此错误：`An I/O error occurred while sending to the backend .`
Greenplum Microsoft SQL Server 数据仓库	Amazon Redshift	解决了用户界面中未显示 `Apply RTRIM to string columns` 选项的问题。
Microsoft SQL Server	适用于 Aurora PostgreSQL 的 Babelfish	增加了对适用于 Aurora PostgreSQL 的 Babelfish 的支持，作为目标平台。用户现在可以创建评估报告，以对从 SQL Server 到适用于 Aurora PostgreSQL 的 Babelfish 的迁移进行估算。
Netezza	Amazon Redshift	改进了项目设置中的优化策略。

来源	目标	新增功能、增强或修复
SAP ASE	Aurora PostgreSQL PostgreSQL	实现了为索引生成唯一名称的功能。
SAP ASE	Aurora PostgreSQL PostgreSQL	修复了目标脚本中索引列重复的问题。
Snowflake	Amazon Redshift	解决了用户界面中未显示隐藏空架构、隐藏空数据库和隐藏系统数据库/架构选项的问题。
Teradata	Amazon Redshift RSQL	增加了对将 Teradata MultiLoad 作业脚本转换为 Amazon Redshift RSQL 脚本的支持。
Teradata	Amazon Redshift RSQL	修复了在 FastLoad 和 FastExport 脚本中转换替换变量时出现的问题。
Teradata	Amazon Redshift RSQL	修复了从摘要选项卡切换后操作项选项卡未显示操作项的问题。
Teradata	Amazon Redshift RSQL	解决了FastExport 脚本转换期间生成报告后出现错误的问题。
Teradata	Amazon Redshift RSQL	解决了 Shell 脚本转换后的格式问题。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift RSQL	修复了一个问题，现在可以在转换后的脚本中注释 AI 13177。
Teradata	Amazon Redshift	修复了时态表转换失败的问题。
Teradata	Amazon Redshift	改进了 `SET QUERY_BAND` 语句的转换。
Teradata	Amazon Redshift	修复了 `NORMALIZE` 操作转换失败的问题。
Vertica	Amazon Redshift	改进了 AI 17008 的描述。

已解决的问题：

- 常规改进。

B AWS SCT uild 655 的发布说明

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift RSQL	修复了在使用 FastLoad 或 MultiLoad时确保所有评估问题都显示在报告中的问题。
Teradata	Amazon Redshift RSQL	增加了对将 Teradata FastExport 作业脚本转换为 Amazon Redshift RSQL 脚本的支持。
Teradata	Amazon Redshift RSQL	修复了使用时确保在离线模式下启用 “将清单保存到 S3” 操作的问题 FastLoad。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift RSQL	修复了一个问题，以确保映射规则适用于类似的脚本 FastLoad。
Greenplum	Amazon Redshift	已将 Greenplum 支持的最低驱动程序版本提高到 42.2.5。
Greenplum	Amazon Redshift	添加了使用驱动程序版本 42.2.5 或更高版本通过 SSL 与 Greenplum 的连接。
Oracle 数据仓库	Amazon Redshift	改进了对在其他用户定义标量函数中执行自定义用户定义标量函数 (UDF) 的支持。
Oracle 数据仓库	Amazon Redshift	修复了函数未应用于数据库并出现此错误的问题：`Failed to compile udf` 。
Oracle 数据仓库	Amazon Redshift	通过使用适当的类型声明（例如，用于 %ROWTYPE 参数的 `pls-type`）改进了转换。
Teradata	Amazon Redshift RSQL	解决了信息类型评估问题未显示在报告中的问题。
Teradata	Amazon Redshift RSQL	在转换某些脚本后解决了转换程序错误。
Teradata	Amazon Redshift RSQL	修复了一个问题，现在可以在转换后的脚本中对问题进行评论。
Teradata	Amazon Redshift	解决了转换后 FastExport ->EXPORT-> “null” 改为 “CAST” 的问题。
Teradata	Amazon Redshift	解决了如果使用驱动程序版本 1.2.43 在使用 `Cause:[JDBC Driver]String index out of range: 0` 应用时应用扩展包的某些函数会失败的问题

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	SET 表转换：插入选择语句添加了 SET 表模拟。
Teradata	Amazon Redshift	CAST：支持其他数据类型强制转换。
Teradata	Amazon Redshift	修复了“other_current_time_01”转换失败的问题
Teradata	Amazon Redshift	Teradata FastExport — Amazon Redshift RSQL：改进 Teradata 命令的转换——字段 FastExport
Teradata	Amazon Redshift	Teradata FastExport — Amazon Redshift RSQL：改进 Teradata 命令的转换——布局 FastExport
Oracle	PostgreSQL Aurora PostgreSQL	解决了重新转换后对象的目标脚本更改且显示 SAVE EXCEPTIONS STATEMENT 的问题。
Oracle	PostgreSQL Aurora PostgreSQL	解决了 `proc_cursor_with_calc_columns` 转换后在 ORDER BY 子句中指定错误字段的问题。
Oracle	PostgreSQL Aurora PostgreSQL	已解决：在 ASSOCIATIVE COLLECTION 转换中需要额外的 `aws_oracle_ext$array_id$temporary` 变量声明。

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL Aurora PostgreSQL	已解决：与同一个表所拥有的索引同名的主键转换错误。

已解决的问题：

- 常规改进。

B AWS SCT uild 654 的发布说明

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL Aurora PostgreSQL	解决了分层查询伪列、PRIOR 列解析错误的问题。
Oracle	PostgreSQL Aurora PostgreSQL	解决了包含斜杠和星号（/*）的多行注释转换不正确的问题。
Oracle	PostgreSQL Aurora PostgreSQL	在扩展包中添加了系统视图 USER_COL_COMMENTS 模拟。

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL Aurora PostgreSQL	改进了引用文字的转换。
DB2 LUW	PostgreSQL Aurora PostgreSQL	改进了在表、视图、别名或列的描述中添加或替换标签的 LABEL 语句的转换。
Oracle	无	用 DBA_USERS 视图替换了 SYS.USER$ 系统表，并改进了查询。
Oracle 数据仓库	Amazon Redshift	更新了 Oracle 数据仓库元数据查询。
Teradata	Amazon Redshift RSQL	增加了对将 shell、Teradata 和 Teradata Basic Teradata FastLoad 查询 (BTEQ) 脚本转换为 Amazon Redshift RSQL 脚本的支持。
Teradata BTEQ	Amazon Redshift RSQL	解决了“merge_01”转换不正确的问题。
Teradata BTEQ	Amazon Redshift RSQL	解决了问题，使结束或识别（EOI）出现在脚本末尾的新行。
Azure Synapse	Amazon Redshift	改进了 Azure Synapse 的密码不正确时出现的错误消息。
Teradata	Amazon Redshift	改进了 UPDATE 语句的转换，以按照 Teradata 标准继续使用正确的别名。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	解决了未收到操作的游标转换错误。
Teradata	Amazon Redshift	解决了 TD_NORMALIZE_OVERLAP 转换会丢掉行的问题。
Teradata	Amazon Redshift	现在支持对增强的 TO_DATE 函数进行严格的日期检查。
Teradata	Amazon Redshift	改进了内置函数 TO_NUMBER(n) 的转换。
Teradata	Amazon Redshift	解决了元数据树中缺少架构类别的问题。
Greenplum	Amazon Redshift	在为 Greenplum 表创建虚拟分区时，在列表中添加了 GP_SEGMENT_ID 选项。
Greenplum	Amazon Redshift	解决了未在目标上应用函数的问题。
MS SQL Server 数据仓库	Amazon Redshift	解决了转换后出现转换错误而没有显示 AI 9996 的问题。
MS SQL Server 数据仓库	Amazon Redshift	解决了打开扩展包向导时记录错误的问题。
MS SQL Server 数据仓库	Amazon Redshift	解决了 Redshift Python 函数使用的注释样式不正确的问题。
Netezza	Amazon Redshift	解决了无法创建带有配置文件的 Netezza—Redshift 扩展包的问题。AWS

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift RSQL	改进了 FastLoad 会话命令的转换。
Teradata	Amazon Redshift RSQL	改进了 FastLoad 脚本评估报告。
Teradata	Amazon Redshift RSQL	实现了 FastLoad WR ITER 保存到 S3 操作。
Teradata	Amazon Redshift RSQL	解决了 FastLoad “保存脚本\ 将清单保存到 s3” 按钮未激活的问题。
Teradata	Amazon Redshift RSQL	解决了 FastLoad multifile_script 在转换后只创建了一个清单文件而不是预期的三个文件的问题。
Teradata	Amazon Redshift RSQL	解决 FastLoad 了 S3 路径中显示多余文件夹的问题。
Teradata	Amazon Redshift RSQL	解决 FastLoad 了 S3 路径中清单文件名称不正确的问题。

已解决的问题：

- 常规改进。

B AWS SCT uild 653 的发布说明

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL Aurora PostgreSQL	实现了转换在被调用函数或过程中创建的动态 SQL 的功能。
Oracle	PostgreSQL Aurora PostgreSQL	改进了动态 SQL 转换：将输入参数作为绑定变量。
Oracle 数据仓库 18、19	Amazon Redshift	改进了 Oracle 到 Redshift 的转换：增强的内置转换功能。聚合 LISTAGG；分析 LISTAGG。
Oracle 数据仓库 18、19	Amazon Redshift	改进了 Oracle 到 Redshift 的转换：查询新功能。
Vertica	Amazon Redshift	改进了 Vertica 到 Redshift 的转换：SSL 到 JDBC 连接，SSL=True。
MS SQL Server 数据仓库	Amazon Redshift	MS SQL Server 到 Redshift 的转换改进：外部表。
Teradata	Amazon Redshift	Teradata 到 Redshift 的转换改进：INTERVAL 数据类型算术运算。
Teradata	Amazon Redshift	Teradata 到 Redshift 的转换改进：支持横向列别名。

来源	目标	新增功能、增强或修复
Oracle	无	现在以下加载程序查询使用 DBA_USERS 代替 SYS.USER$ ： • get-tree-path-list-by-name-path.sql • estimate-table-or-view-constraints-by-schema.sql • estimate-table-or-view-constraints-by-selected-schemas.sql
Teradata	Amazon Redshift	改进了 SCT 将 Teradata 宏转换为 Redshift 存储过程时的注释对齐方式。
Oracle 数据仓库	Amazon Redshift	改进了日期/时间戳格式元素的转换：TO_DATE、TO_TIMESTAMP 和 TO_TIMESTAMP_TZ
Teradata	Amazon Redshift	解决了 Teradata 游标转换错误。
Teradata	Amazon Redshift	解决了导致转换期间 TD_NORMALIZE_OVERLAP 的属性被删除的问题。
Teradata	Amazon Redshift	解决了 SCT 转换查询时忽略 MAX 函数的问题。
Teradata	Amazon Redshift	SCT 现在将 Teradata CHARACTERS 函数转换为 Redshift LENGTH 函数。
Teradata	Amazon Redshift	对于最常用的格式，SCT 现在支持将 FORMAT 转换为 TO_CHAR。
全部	全部	改进了加密例程的转换。

已解决的问题：

- 常规改进。

B AWS SCT uild 652 的发布说明

来源	目标	新增功能、增强或修复
Microsoft SQL Server	PostgreSQL	为 `sp_getapplock` 和 `sp_releaseapplock` 函数添加了应用程序锁定。
无	Amazon Redshift	命令行界面（CLI）改进：实现了脚本命令模式。
Oracle	PostgreSQL Aurora PostgreSQL	在动态 SQL 中实现了例程参数采样。
Oracle	PostgreSQL Aurora PostgreSQL	改进了在调用函数或过程中创建的动态 SQL 的转换。
Microsoft SQL Server Oracle DB2 LUW	Aurora PostgreSQL	每个 lambda 函数仅通过策略部署和配置一次，并且所有可能的源都可重复使用常见的 lambda 函数。
DB2 LUW	PostgreSQL	解决了在使用 DB2 LUW 作为源时导致错误消息“9996（严重性：重大）出现转换程序错误”的问题。
Teradata	Amazon Redshift	即将发布的 Amazon Redshift 支持递归表表达式。

来源	目标	新增功能、增强或修复
Azure Synapse	Amazon Redshift	实施了架构优化规则。
Teradata	Amazon Redshift	支持从 Teradata 宏到 Redshift 存储过程的时区转换。
Teradata	Amazon Redshift	支持对周期值进行算术运算。
Teradata	Amazon Redshift	支持 Teradata 递归公用表表达式（RECURSIVE CTE）的转换。
Teradata	Amazon Redshift	通过用户设置 `enable_case_sensitive_identifier` 支持区分大小写的标识符。因此，“COLUMN_NAME”和“Column_Name”变成了不同的列名。
Teradata	Amazon Redshift	解决了十进制数据类型问题，使十进制字段的转换精度相同。
Teradata	Amazon Redshift	解决了间隔算术转换问题，以便间隔算术减法正确转换。
Teradata	Amazon Redshift	改进了 Teradata NUMBER 到 DATE 类型强制转换。
Teradata	Amazon Redshift	改进了 Teradata DATE 到 NUMBER 类型的强制转换
Teradata BTEQ	Amazon Redshift	改进了 PERIOD 数据类型转换。
Teradata	Amazon Redshift	解决了为带有 GEOMETRY 列的表加载元数据的问题，使其现在可以从 Teradata 正确加载。
Teradata	Amazon Redshift	在将 Teradata 宏转换为 Redshift 存储过程时，支持合并语句的转换。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	改进了从 Teradata 迁移到 Redshift 时简单宏的转换。
Teradata	Amazon Redshift	已确保 Teradata UPDATE 语句的转换按照 Teradata 标准继续使用正确的别名。

已解决的问题：

- 常规改进。

B AWS SCT uild 651 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	增强了 AWS SCT 报告，用于更新列出的推荐转化操作项的链接。
MS SQL Server	PostgreSQL	增加了对 `STR()` 函数转换的支持。
MS SQL Server	PostgreSQL	支持将按位 EXOR 运算符（Microsoft SQL Server 中的 ^）转换为 PostgreSQL 作为 # 运算符。
Oracle	PostgreSQL	解决了 AWS SCT 扩展包`aws_oracle_ext.UNISTR(null)` 功能在 PostgreSQL NULL 目标上挂起的问题。 AWS SCT 现在可以处理NULL。
Teradata BTEQ	Amazon Redshift RSQL	改进了转换，以解决转换 Amazon Redshift RSQL MERGE 时出现转换错误的问题。
Oracle 数据仓库	Amazon Redshift	实现了增强的内置函数。

来源	目标	新增功能、增强或修复
Oracle 数据仓库	Amazon Redshift	添加了元数据功能驱动的增强功能，包括自动列表分区（TBL_PART_LIST_AUTO）、多列列表（TBL_PART_MULTI_LIST）和间隔引用（TBL_PART_RANGE_INTVAL_REF）。
none	Amazon Redshift	增加了用于 `UNION ALL` 转换的物理分区的分区表限制。
Teradata	Amazon Redshift	对评估报告的范围进行了转换改进。
Teradata	Amazon Redshift	改进了复杂的 Teradata 宏转换。
Teradata	Amazon Redshift	改进了 Teradata 宏到 Amazon Redshift 存储过程的转换，同时注释掉不支持的 SQL。
Teradata	Amazon Redshift	解决了将 Teradata 宏转换为 Amazon Redshift 存储过程导致别名引用错误的问题。
Teradata	Amazon Redshift	改进了 Teradata `QUALIFY` 语句的转换。
Teradata	Amazon Redshift	改进了转化，可将注释转发到 Amazon Redshift，并保留对视图执行的更改历史记录。
Teradata	Amazon Redshift	解决了 RESET WHEN 子句未导致正确转换的问题。
Teradata BTEQ	Amazon Redshift	改进了包含 MERGE 语句的 BTEQ 脚本转换。
Teradata	Amazon Redshift	添加了用于改进 PERIOD 数据类型字段转换的内置函数。
Microsoft SQL Server	Amazon Redshift	增强了 TIME 数据类型的转换数据类型映射功能。

来源	目标	新增功能、增强或修复
全部	全部	增加了对初次发布的 PDF 格式的 AWS Schema Conversion Tool CLI 参考手册的访问权限。请参阅 AWS Schema Conversion Tool CLI 参考。

已解决的问题：

- 常规改进。

B AWS SCT uild 650 的发布说明

来源	目标	新增功能、增强或修复
全部	全部	更新并增强了提取代理的使用，包括： • 一种与共享存储和专用复制代理配合使用的配置。 • 将数据提取任务从一个项目导出和导入到另一个项目。 • 支持将 Azure SQL 数据仓库（Azure Synapse）作为源。 • 使用原生 Netezza 分区。 有关更多信息，请参阅 将本地数据仓库中的数据迁移到 Amazon Redshift。
全部	Amazon RDS PostgreSQL 13	AWS SCT 现在支持 Amazon RDS PostgreSQL 13 作为目标。
Microsoft SQL Server	Aurora PostgreSQL	改进了结果集从 Microsoft SQL Server 过程到 Aurora PostgreSQL 目标的转换。

来源	目标	新增功能、增强或修复
Oracle 数据仓库	Amazon Redshift	改进了 Oracle 到 Amazon Redshift 的转换。
Oracle 数据仓库	Amazon Redshift	改进了动态 SQL 语句的转换。
Oracle 数据仓库	Amazon Redshift	改进了 SQL 用户定义函数的转换。
Oracle 数据仓库	Amazon Redshift	澄清了 AWS SCT 不支持转换外部表的消息。
Oracle 数据仓库	Amazon Redshift	增强的内置转换函数。
Teradata BTEQ	Amazon Redshift RSQL	改进了使用 AWS SCT GUI 时在 BTEQ 脚本中处理替换参数。
Microsoft SQL Server 数据仓库 Microsoft SQL Server Azure Azure Synapse	全部	升级了 Microsoft SQL Server、Azure、Azure Synapse 支持的最低的 JDBC 驱动程序版本。

已解决的问题：

- Teradata：宏转换的其他改进[已解决]
- 特殊字符在目标中转义导致 SQL 错误，需要重新放回原处[已解决]

- 常规改进

B AWS SCT uild 649 的发布说明

来源	目标	新增功能、增强或修复
Microsoft SQL Server 数据仓库	Amazon Redshift	改进了 MSSQL 到 Amazon Redshift 的转换，以支持时态表。
Oracle 数据仓库	Amazon Redshift	增强了内置函数功能，例如： 转换函数 • TO_BINARY_DOUBLE • TO_BINARY_FLOAT • TO_NUMBER • TO_DATE • TO_TIMESTAMP • TO_TIMESTAMP_TZ • TO_DSINTERVAL • TO_YMINTERVAL • VALIDATE_CONVERSION
Oracle 数据仓库	Amazon Redshift	实现了近似查询处理函数增强，例如： 聚合函数 •

来源	目标	新增功能、增强或修复
		ANY_VALUE • APPROX_COUNT_DISTINCT • APPROX_COUNT_DISTINCT_DETAIL • APPROX_COUNT_DISTINCT_AGG • LISTAGG • TO_APPROX_COUNT_DISTINCT
Teradata	Amazon Redshift	增强了 Teradata 自动排序键和分配键选择的转换。数据库引擎会自动选择分配键和排序键。在当前项目设置 > 优化策略 > 初始键选择策略对话框中引入了一个标有使用 Amazon Redshift 自动调整表格的单选按钮。
Teradata	Amazon Redshift	增强的 AWS SCT 表加载器可确保从 Teradata AWS SCT 加载所有表。
Teradata	Amazon Redshift	增强了转换功能，以便 Amazon Redshift 支持相关的子查询模式，其中包括一个简单的 WHERE NOT EXISTS 子句。
Teradata	Amazon Redshift	支持在宏中使用 ECHO 命令。
DB2 LUW	PostgreSQL Aurora PostgreSQL	支持 DYNAMIC RESULTS SETS 转换，包括： • 游标子句 WITH RETURN/WITH RETURN TO CLIENT • DYNAMIC RESULT SETS 例程子句转换

来源	目标	新增功能、增强或修复
Microsoft SQL Server Oracle DB2 LUW SAP ASE	Aurora PostgreSQL	支持将对当前 Aurora RDS PostgreSQL 作为目标。
Microsoft SQL Server Oracle DB2 LUW SAP ASE	MariaDB	支持将 MariaDB 10.5 作为目标。
Microsoft SQL Server	MariaDB	实现了对 INSERT-RETURNING 的支持，它将返回插入行的结果集。
Oracle	Aurora PostgreSQL	支持 XMLFOREST 函数，用于从 Oracle 转换到 Aurora PostgreSQL。

已解决的问题：

- 常规改进。

B AWS SCT uild 648 的发布说明

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL Amazon Aurora PostgreSQL 兼容版	实现了 Aurora PostgreSQL 扩展包自定义应用模式：数字/日期和文本类型的运算符。
Oracle Microsoft SQL Server DB2 LUW	Aurora PostgreSQL	实现了 Aurora PostgreSQL Lambda 调用配置：创建 aws_lambda 扩展程序；向 Aurora PostgreSQL 集群分配 IAM 角色。 • Oracle-电子邮件、作业、队列、文件 WebAgent • DB2：电子邮件、任务、文件 • Microsoft SQL Server：电子邮件、代理
Oracle	PostgreSQL	实现了 FORALL 语句转换重构： • FORALL 语句 • FORALL ... SAVE EXCEPTIONS • 包含 BULK COLLECT 的 RETURNING INTO • SQL%BULK_EXCEPTIONS 系统集合
Oracle 数据仓库 18、19	Amazon Redshift	Oracle 到 Amazon Redshift 的转换改进：增强转换内置函数的功能。聚合 LISTAGG；分析 LISTAGG。

来源	目标	新增功能、增强或修复
Oracle 数据仓库 18、19	Amazon Redshift	Oracle 到 Amazon Redshift 的转换改进：查询新功能。
Vertica	Amazon Redshift	Vertica 到 Amazon Redshift 的转换改进：SSL 到 JDBC 连接，SSL=True。
Microsoft SQL Server 数据仓库	Amazon Redshift	Microsoft SQL Server 到 Redshift 的转换改进：外部表。
Teradata	Amazon Redshift	Teradata 到 Redshift 的转换改进：INTERVAL 数据类型算术运算。
Teradata	Amazon Redshift	Teradata 到 Redshift 的转换改进：支持横向列别名。

已解决的问题：

- 常规改进

B AWS SCT uild 647 的发布说明

来源	目标	新增功能、增强或修复
Microsoft SQL Server	Microsoft SQL Server	RDS 现在支持数据库邮件功能。
Microsoft SQL Server	MySQL	每种类型标识符名称的最大长度：SQL Server 中对象名称（例如表、约束、列）的最大长度为 128 个字符。MySQL 中对象名称的最大长度为 64 个字符。要将转换后的对象写入 MySQL 数据库，您需要缩短其名称。为防止剪切后出现重复名称，您需要在新名称中添加原始对象名称的“校验和”。

来源	目标	新增功能、增强或修复

		按如下方式剪切长度超过 64 个字符的名称：
		`[first N chars]() + "" + [checksum]()`
		`[first N chars] = 64 - 1 - [length of checksum string]`
		例如：
		`example_of_a_test_schema_with_a_name_length_greater_than_64_characters ?? example_of_a_test_schema_with_a_name_length_greater_than_64_9703`

Oracle	MySQL/Aurora MySQL	实现了存储对象注释的加载和转换。例如，处理表格上的注释以及处理表格/视图列上的注释。
Teradata	Amazon Redshift	支持 TIME 数据类型转换。
Teradata	Amazon Redshift	转换改进：实现了 TD_NORMALIZE_OVERLAP。
Microsoft SQL Server 数据仓库	Amazon Redshift	转换改进：带 WITH 子句的 SELECT；不包含 FROM 的 SELECT
全部	全部	AWS SCT 数据迁移服务评估员 (DMSA) — 这项新功能使您能够评估多台服务器并收到一份摘要报告，该报告显示了您的环境的最佳目标方向。
全部	全部	AWS SCT 向导-目标比较现在可以在单个表格视图中显示目标之间的差异。
全部	全部	树筛选器用户界面：重新设计的元数据筛选器可处理更复杂的筛选模式。

来源	目标	新增功能、增强或修复
全部	全部	评估报告：重新设计的警告部分提供了对问题的更好描述和更清晰的理解。

已解决的问题：

- 常规改进
- 数据提取器-子任务失败， ConcurrentModificationException [已解决]。
- Microsoft SQL Server 到 MySQL：最大标识符长度[已解决]。

B AWS SCT uild 646 的发布说明

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL	改进了 TM 格式模型的实现。
Oracle	PostgreSQL	SP 格式掩码仅为英语提供对 SP 后缀的基本支持。
Oracle	PostgreSQL	Oracle 长对象名称处理 — AWS SCT 现在根据目标最大标识符长度属性处理 Oracle 长对象名称。
	Amazon Redshift	使用 Amazon Redshift 编码 AZ64 AWS SCT — 为某些数据类型添加了压缩编码 AZ64
Teradata	Amazon Redshift	增加了对隐式事务转换的支持。
Teradata	Amazon Redshift	增加了对 Teradata 地理空间内置函数：`ST_LineString` 方法
Greenplum	Amazon Redshift	Greenplum 序列转换：在“属性”选项卡中添加了以下项目：最小值、最大值、增量、周期。

来源	目标	新增功能、增强或修复
Greenplum	Amazon Redshift	解析器：添加了“char”数据类型解析。
Greenplum	Amazon Redshift	字符转换长度：更新了字符类型的 PL/pgSQL 转换。
Greenplum	Amazon Redshift	解决了 Greenplum 分配密钥选择的问题，即表具有分配密钥但 AWS SCT 无法识别和获取随机分布的表。
Teradata	Amazon Redshift	Teradata 游标支持：增加了对游标转换的支持。
Teradata	Amazon Redshift	身份列：添加了对身份列转换的支持。
Teradata	Amazon Redshift	INTERVAL 数据类型：添加了对 INTERVAL 数据类型转换的支持。

已解决的问题：

- 常规改进
- Greenplum：由于日志中存在错误，无法运行转换[已解决]。
- MSSQL：PostgreSQL：转换 LAG 函数时出现转换程序错误[已解决]。
- MSSQL：PostgreSQL: SCOPE_IDENTITY [已解决]。
- AWS SCT 在 DW 项目中悬而未决 [已解决]。
- 需要映射规则来删除 AWS SCT [已解决] 中列名上的额外空格。

B AWS SCT uild 645 的发布说明

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	提供解决方案来解析 Teradata 非完全限定视图（视图名称或视图中的非完全限定对象）。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	为计算节点添加了对 ASCII 函数的支持。
Teradata	Amazon Redshift	当在CHAR定义为的 Teradata 中 AWS SCT 发现多字节数据时CHAR(N)，该数据将转换为 Amazon VARCHAR(3*N) Redshift。
Teradata	Amazon Redshift	提供日期和数字之间的 Teradata CAST 转换。 • SELECT Cast('2020-07-17' AS BIGINT) • SELECT Cast(20200630 - 19000000 AS DATE)
Teradata	Amazon Redshift	支持将 Teradata PERIOD 数据类型转换为两个 Amazon Redshift TIMESTAMP 列： • PERIOD(TIMESTAMP) • PERIOD(TIMESTAMP WITH TIMEZONE)
Teradata	Amazon Redshift	支持使用 RESET WHEN 子句转换 Teradata RANK 函数。
Teradata	Amazon Redshift	改进了在显式数据类型转换中对 CAST 的支持，以及对表达式的隐式 CAST 的支持。
Teradata	Amazon Redshift	报告不支持的关联子查询模式。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的关联的子查询。
none	Amazon Redshift	改进了对 RA3 节点类型的表限制支持。
Teradata	Amazon Redshift	增加了对 Teradata 地理空间数据提取的支持。有关更多信息，请参阅《Amazon Redshift 数据库开发人员指南》中的在 Amazon Redshift 中查询空间数据。

来源	目标	新增功能、增强或修复
Microsoft SQL Server	PostgreSQL	添加了选项 convert_procedures_to_function 。

已解决的问题：

- 常规改进

B AWS SCT uild 644 的发布说明

1.0.643 AWS SCT 版本的更改已合并到 1.0.644 版本中 AWS SCT 。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	多项转化改进。 • 改进了使用表别名的 QUALIFY 的转换。 • 改进了 IN 运算符的转换。 • 改进了 LIKE 运算符的转换。 • 通过突出显示转换后的代码中的问题改进了转换。 • 改进了 SQL 中的 WHERE、QUALIFY 子句非常态顺序的转换。 • 修复了在程序 UPD_FT_SVC_TRANS_BH_CBH_IND 的 JOIN() 结构转换期间发生的转换程序错误。 • 改进了宏到存储过程的转换。 添加了特殊的 AWS SCT CLI 命令，这些命令可以解析提供的 sql/bteq 脚本并生成有关源代码中遇到的语法结构数量的报告。

来源	目标	新增功能、增强或修复
		• BTEQ 命令的计数 • 处理程序计数 • CAST 案例计数 • DML/DDL 案例计数 • 可更新视图上的 DML 计数 添加了评估报告操作项：Amazon Redshift 不支持自定义日期格式的 Teradata 列。
Oracle	PostgreSQL/Aurora PostgreSQL	添加了保存扩展包安装脚本的功能。 更改了 AI 5334 的严重性级别。 提高了使用记录作为包变量 `IMPLEMENTATION` 的性能。 增加了 `XMLAGG` 聚合函数支持
IBM Db2	PostgreSQL/Aurora PostgreSQL	添加了有关存储对象实施的注释的加载和转换。
MS SQL 数据仓库	Amazon Redshift	转换改进：已解决 `PATINDEX` 问题。 用户界面改进： • 另存为 SQL 以实现源代码树。 • 为多文件脚本的生成添加了附加逻辑。
Vertica	Amazon Redshift	用户界面改进：另存为 SQL 以实现源代码树。

已解决的问题：

- 总体改进 Teradata 和 Amazon Redshift 之间的转换
- 一般错误修复和用户界面改进

B AWS SCT uild 642 的发布说明

1.0.642 AWS Schema Conversion Tool 版本的更改。

> **Note**
>
> AWS Schema Conversion Tool (AWS SCT) build 1.0.642 更改适用于 Windows、Ubuntu 和 Fedora。没有适用于 macOS 的 1.0.642 版本。

来源	目标	新增功能、增强或修复
Microsoft SSIS	AWS Glue	实现了将微软 SQL Server 集成服务 (SSIS) ETL 包转换为。 AWS Glue有关更多信息，请参阅 用 AWS SCT 将 SSIS 转换为 AWS Glue。
Oracle	MariaDB/SQL 模式 =ORACLE /MySQL/ Amazon Aurora MySQL	在 WITH 子句中实现了 PL/SQL 声明部分。
Oracle	PostgreSQL/Aurora PostgreSQL	添加了对 `DBMS_SESSION.RESET_PACKAGE` 和 `DBMS_SESSION.MODIFY_PACKAGE` 的支持。
Vertica	Amazon Redshift	允许将 SQL 脚本从 Vertica 数据库导出到 Amazon Redshift。

已解决的问题：

- 评估报告增强功能。
- 评估报告用户界面增强功能。
- 添加从用户界面更改 JVM 设置的功能。
- 常规改进。

AWS SCT 版本 641 的发布说明

1.0.641 AWS Schema Conversion Tool 版本的更改。

> **Note**
>
> AWS Schema Conversion Tool (AWS SCT) build 1.0.641 更改适用于 Windows、Ubuntu 和 Fedora。没有适用于 macOS 的 1.0.641 版本。

来源	目标	新增功能、增强或修复
Oracle/ MS SQL/ MySQL/ PostgreSQL/Db2 LUW	全部	在.csv 文件中生成时间报告计算结果。
Teradata	Amazon Redshift	增加了对 CSUM 函数的支持。 增加了对 Teradata 地理空间数据类型的支持。
Teradata	全部	添加了对转换身份列的支持。
Greenplum	Amazon Redshift	添加了在 Greenplum 表转换期间对分配方式 AUTO 的支持。
SAP ASE	全部	在.csv 文件中生成时间报告计算结果。

已解决：

- 各种错误修复。
- 各种性能改进。

B AWS SCT uild 640 的发布说明

1.0.633、1.0.634、1.0.635、1.0.636、1.0.637、1.0.637、1.0.638、1.0.639 和 1.0.640 AWS SCT 版本的更改已合并到 1.0.640 版本中。 AWS SCT

> **Note**
>
> AWS SCT build 1.0.640 更改适用于 Windows、Ubuntu 和 Fedora。这些更改不适用于 macOS。
> 你无法在苹果 mac AWS SCT OS 上安装 1.0.640 或更高版本。 AWS SCT 版本 1.0.632 是最后一个支持在苹果 macOS 上安装的版本。

在下表中，您可以找到 AWS Schema Conversion Tool 版本的功能和错误修复（它们已并入 1.0.640 版本）的列表。这些表按源引擎对功能和错误修复进行分组。

主题

版本 1.0.640 Oracle 更改

下表列出了内部版本 1.0.640 更改，其中 Oracle 是源引擎。

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL Aurora PostgreSQL	在 Java 和 Pro*C 应用程序中实现了 SQL 代码转换。
Oracle	PostgreSQL Aurora PostgreSQL	改进了以下函数在 WHERE 子句中使用时的性能： • aws_oracle_ext.to_date • aws_oracle_ext.to_char • aws_oracle_ext.to_number • aws_oracle_ext.sysdate • aws_oracle_ext.sys_context
Oracle	RDS MariaDB 10.4	增加了对所有在线事务处理 (OLTP) 供应商的 RDS MariaDB 10.4 支持。
Oracle	PostgreSQL/Aurora PostgreSQL	增加了对 DBMS_UTILITY.GET_TIME 的支持。 增加了以下模拟： • DBMS_UTILITY.GET_TIME • DBMS_UTILITY.FORMAT_CALL_STACK • DBMS_UTILITY.CURRENT_INSTANCE
Oracle	MariaDB/MySQL/Aurora MySQL/Microsoft SQL Server 模式=Oracle	增加了对 TABLE(DATA,EXTENDED DATA)、VIEW(DATA,EXTENDED DATA) 和 SEQUENCE(DATA) 的支持

来源	目标	新增功能、增强或修复
	/PostgreSQL/Aurora PostgreSQL/RDS Oracle	
Oracle	PostgreSQL/Aurora PostgreSQL/Oracle RDS	可以扩展列的 DEFAULT 定义，以便为显式 NULL 插入应用 DEFAULT。 DEFAULT 子句具有新的 ON NULL 子句。此新子句指示数据库在 INSERT 语句尝试分配计算结果为 NULL 的值时分配指定的默认列值。
Oracle	MariaDB/MariaDB (SQL MODE=ORACLE)	增加了对“身份列”的支持，该列在插入时自动增加。
全部	全部	从 JDK 8 升级到 Amazon Corretto JDK 11。有关更多信息，包括下载链接，请参阅 Amazon Corretto 11 用户指南中的什么是 Amazon Corretto 11？
全部	全部	在评估报告中增加了有关用户数据库中可能存在的不一致之处的信息。

来源	目标	新增功能、增强或修复
Oracle	MariaDB 10.2/MariaDB 10.3/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	DEFAULT 子句具有一个新的 ON NULL 子句，该子句指示数据库在 INSERT 语句尝试分配计算结果为 NULL 的值时分配指定的默认列值。
Oracle	Oracle RDS/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	增加了对 IDENTITY 列的支持。
Oracle	MySQL 8.x	增加了对 CHECK 约束的支持。

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL/Aurora PostgreSQL	实现了使用扩展包例程检查 ANYDATA IS NULL/IS NOT NULL。 已基于 XMLSequence 的 TABLE 函数实现对查询中使用的 VALUE 函数的模拟。 增加了对以下内置例程的 DBMS_LOB 支持： • DBMS_LOB.CREATETEMPORARY • DBMS_LOB.FREETEMPORARY • DBMS_LOB.APPEND
全部	SQL Server	SQL Server 2019：增加了对新索引属性 OPTIMIZE_FOR_SEQUENTIAL_KEY 的支持。 SQL Server 2017：增加了对图形数据库节点和边缘表类型的支持。 SQL Server 2016：增加了对 TEMPORAL TABLES 的支持。
全部	全部	实现了使用虚拟分区覆盖物理分区的功能。数据仓库提取器根据创建的虚拟分区提取数据。
Oracle	Amazon Redshift	在嵌套块中实现了游标属性的转换。 Amazon Redshift 不支持集合。相关变量将转换为 VARCHAR。除将一个变量分配给另一个变量之外的所有集合操作都将被拒绝，包括启动和集合元素访问。 实现了 Amazon Redshift 分配方式 = 自动。

来源	目标	新增功能、增强或修复
Oracle	PostgreSQL/Aurora PostgreSQL	如果在 PostgreSQL 中保留了 Oracle 中的非保留字，则会出现以下情况： • 如果此非保留字带引号，则其大小写形式将保留，并保留引号。 • 如果此非保留字不带引号，则它将被转换为大写形式并加上引号。 已实现使用函数作为 LTRIM、RTRIM 和 TRIM 函数的输入的能力。 SELECT DISTINCT, ORDER BY 表达式必须显示在选择列表中。 对于紧随其后的带有默认值的参数的游标参数， AWS SCT 添加 DEFAULT IS NULL 子句 源 OUT 游标参数将转换为 IN 游标参数。 已通过在“转换设置”下添加“包变量逻辑实现”选项来重新实现包变量。可用的设置为：“会话变量”和“plv8 全局对象”。默认值为“会话变量”。 实现了带 dblink 和 pg_background 的 AUTONOMOUS_TRANSACTION pragma 支持。
Oracle	全部	实现了视图 SYS_%_TAB_COMMENTS。
Oracle	PostgreSQL	PostgreSQL 中不支持筛选器的变量输入。在从 Oracle 转换到 PostgreSQL 时，如果遇到变量筛选器，系统现在会报告异常。

来源	目标	新增功能、增强或修复
Oracle	Amazon Redshift	实现了存储代码 FOR..LOOP 游标转换改进。 已使用默认参数实现函数/过程的存储代码调用。 实现了可在没有 WHERE 子句的情况下使用别名执行 UPDATE 操作的存储代码。 实现了使用 SELECT FOME Dual 预制其他情形的存储代码函数。 实现了存储代码 Table%ROWTYPE 参数和包变量。 实现了用于 JAVA 和外部过程的存储代码。 在存储代码中实现了标准 Oracle 包。

版本 1.0.640 Microsoft SQL Server 更改

下表列出了内部版本 1.0.640 更改，其中 Microsoft SQL Server 是源引擎。

来源	目标	新增功能、增强或修复
Microsoft Azure/Microsoft SQL Server	PostgreSQL/Aurora PostgreSQL/MySQL/Aurora MySQL	增加了对 COLUMN STORE 索引的支持。
Microsoft SQL Server	RDS MariaDB 10.4	增加了对所有在线事务处理 (OLTP) 供应商的 RDS MariaDB 10.4 支持。
Azure/SQL Server	MariaDB/MySQL/Aurora MySQL/	增加了对 OPTIMIZE_FOR_SEQUENTIAL_KEY 索引属性的支持。

来源	目标	新增功能、增强或修复
	PostgreSQL/Aurora PostgreSQL	
Azure/SQL Server	MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	增加了对数据库节点和边缘表类型的支持。
Azure/SQL Server	MariaDB/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	增加了对 TEMPORAL TABLES 的支持。
全部	全部	从 JDK 8 升级到 Amazon Corretto JDK 11。有关更多信息，包括下载链接，请参阅 Amazon Corretto 11 用户指南中的什么是 Amazon Corretto 11？
全部	全部	在评估报告中增加了有关用户数据库中可能存在的不一致之处的信息。

来源	目标	新增功能、增强或修复
Azure/SQL Server	MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL/MariaDB	增加了对 SQL Server 图形架构的 DML 处理的支持。
SQL Server	Aurora PostgreSQL	增加了用于转换不带 `par_` 前缀的参数的选项。
Azure/SQL Server	MySQL 8.x	增加了对 CHECK 约束的支持。
全部	SQL Server	SQL Server 2019：增加了对新索引属性 OPTIMIZE_FOR_SEQUENTIAL_KEY 的支持。 SQL Server 2017：增加了对图形数据库节点和边缘表类型的支持。 SQL Server 2016：增加了对 TEMPORAL TABLES 的支持。
全部	全部	实现了使用虚拟分区覆盖物理分区的功能。数据仓库提取器根据创建的虚拟分区提取数据。
SQL Server	AWS Glue（Python 外壳）	转换改进，包括： • 实现了到 Python.String 的内置函数转换。 • 在存储代码中实现了 EXECUTE 和 EXEC。 • 已使用表类型实现。

来源	目标	新增功能、增强或修复
Azure/SQL Server	PostgreSQL/Aurora PostgreSQL	已将 $TMP 过程设为可选。
SQL Server	MySQL/Aurora MySQL	带日期的扩展算术运算。 构造模拟 TOP (表达式) WITH TIES。 在调用带生成的 refcursor 的过程后，refcursor 现在将关闭。 无法在 Aurora MySQL 中设置 GLOBAL 隔离级别。只能更改会话范围。事务的默认行为是使用 REPEATABLE READ 和一致性读取。可能需要修改设计用于 READ COMMITTED 的应用程序。或者，它们可以明确地将默认值更改为 READ COMMITTED。
SQL Server	AWS Glue（Python 外壳）	SQL Server 语句生成一个完整的结果集，但有时，最佳做法是一次处理一行结果。在结果集上打开游标将允许一次处理结果集中的一行结果。可以将游标分配给具有游标数据类型的变量或参数。 实现了为存储代码封装一系列 Transact-SQL 语句，即使 Python 不支持将 SQL Server 的 BEGIN 和 END 作为控制流，也可以运行一组 Transact-SQL 语句。 不支持 SQL Server LABEL 和 GOTO 语句 AWS Glue。如果 AWS SCT 在代码中遇到标签，则会跳过该标签。如果 AWS SCT 遇到 GOTO 语句，则会对该语句进行注释。

来源	目标	新增功能、增强或修复
SQL Server	Amazon Redshift	通过实现 IF... ELSE 控制为存储代码实现了 Transact-SQL 语句的条件 处理。 实现了为存储代码封装一系列 Transact-SQL 语句，以便将一组 Transact-SQL 语句作为一个语句块运行。支持的嵌套 BEGIN ... END 块。 在存储代码中实现了 SET 和 SELECT。 通过在表上创建用户指定的排序键，在 Amazon Redshift （不支持索引）中实现了 CREATE INDEX。

版本 1.0.640 MySQL 更改

下表列出了内部版本 1.0.640 更改，其中 MySQL 是源引擎。

来源	目标	新增功能、增强或修复
MySQL	PostgreSQL 12.x	增加了对生成的列的支持。
全部	全部	从 JDK 8 升级到 Amazon Corretto JDK 11。有关更多信息，包括下载链接，请参阅 Amazon Corretto 11 用户指南中的什么是 Amazon Corretto 11？
全部	全部	在评估报告中增加了有关用户数据库中可能存在的不一致之处的信息。
MySQL	PostgreSQL/Aurora PostgreSQL 11。	增加了对以下内容的支持： • SQL 存储过程中的嵌入式事务。 • 能够对存储过程执行 CALL SQL。 • 能够创建 SQL 存储过程。

来源	目标	新增功能、增强或修复
全部	SQL Server	SQL Server 2019：增加了对新索引属性 OPTIMIZE_FOR_SEQUENTIAL_KEY 的支持。 SQL Server 2017：增加了对图形数据库节点和边缘表类型的支持。 SQL Server 2016：增加了对 TEMPORAL TABLES 的支持。
全部	全部	实现了使用虚拟分区覆盖物理分区的功能。数据仓库提取器根据创建的虚拟分区提取数据。

版本 1.0.640 PostgreSQL 更改

下表列出了内部版本 1.0.640 更改，其中 PostgreSQL 是源引擎。

来源	目标	新增功能、增强或修复
PostgreSQL	MySQL 8.x	MySQL 现在支持创建功能索引键部分，该部分索引表达式值而不是列值。功能键部分可以索引其他方式无法索引的值，例如 JSON 值。 MySQL 现在支持 Now CTE 和递归 CTE。
全部	全部	从 JDK 8 升级到 Amazon Corretto JDK 11。有关更多信息，包括下载链接，请参阅 Amazon Corretto 11 用户指南中的什么是 Amazon Corretto 11？
全部	全部	在评估报告中增加了有关用户数据库中可能存在的不一致之处的信息。
PostgreSQL 11.x	PostgreSQL/Aurora PostgreSQL 11。	增加了对以下内容的支持： • SQL 存储过程中的嵌入式事务。 • 能够对存储过程执行 CALL SQL。 • 能够创建 SQL 存储过程。

来源	目标	新增功能、增强或修复
PostgreSQL	MySQL 8.x	增加了对降序索引的 MySQL 支持。不再忽略索引定义中的 DESC，而是导致按照降序顺序存储键值。 增加了对将表达式用作数据类型规范中的默认值的 MySQL 支持，包含表达式作为 BLOB、TEXT、GEOMETRY 和 JSON 数据类型的默认值。 多个现有的聚合函数现在可用作窗口函数： • 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 支持窗口函数，对于查询的每一行，这些函数使用与该行相关的行执行计算。 • CUME_DIST()

来源	目标	新增功能、增强或修复
		• DENSE_RANK() • FIRST_VALUE() • LAG() • LAST_VALUE() • LEAD() • NTH_VALUE() • NTILE() • PERCENT_RANK() • RANK() • ROW_NUMBER()
PostgreSQL	MySQL 8.x	增加了对 CHECK 约束的支持。
全部	SQL Server	SQL Server 2019：增加了对新索引属性 OPTIMIZE_FOR_SEQUENTIAL_KEY 的支持。 SQL Server 2017：增加了对图形数据库节点和边缘表类型的支持。 SQL Server 2016：增加了对 TEMPORAL TABLES 的支持。
全部	全部	实现了使用虚拟分区覆盖物理分区的功能。数据仓库提取器根据创建的虚拟分区提取数据。
PostgreSQL/Aurora PostgreSQL	全部	增加了系统视图 sysindexes 模拟。 如果在没有指定 INTO 的过程中存在 SELECT 语句，则会为目标上的过程创建类型 refcursor 的参数 INOUT p_refcur。

版本 1.0.640 Db2 LUW 更改

下表列出了内部版本 1.0.640 更改，其中 DB2 LUW 是源引擎。

来源	目标	新增功能、增强或修复
DB2 LUW	RDS MariaDB 10.4	增加了对所有在线事务处理 (OLTP) 供应商的 RDS MariaDB 10.4 支持。
全部	全部	从 JDK 8 升级到 Amazon Corretto JDK 11。有关更多信息，包括下载链接，请参阅 Amazon Corretto 11 用户指南中的什么是 Amazon Corretto 11？
全部	全部	在评估报告中增加了有关用户数据库中可能存在的不一致之处的信息。
DB2 LUW	MySQL 8.0.17	增加了 CHECK 约束支持。
全部	SQL Server	SQL Server 2019：增加了对新索引属性 OPTIMIZE_FOR_SEQUENTIAL_KEY 的支持。 SQL Server 2017：增加了对图形数据库节点和边缘表类型的支持。 SQL Server 2016：增加了对 TEMPORAL TABLES 的支持。
全部	全部	实现了使用虚拟分区覆盖物理分区的功能。数据仓库提取器根据创建的虚拟分区提取数据。

版本 1.0.640 Teradata 更改

下表列出了面向 Teradata 源引擎的内部版本 1.0.640 更改。

来源	目标	新增功能、增强或修复
Teradata	Amazon Redshift	增加了对 MERGE 和 QUALIFY 语句的支持。 已从 Teradata 语句中删除 LOCKING ROWS FOR ACCESS 子句。 增加了对 CAST 函数的支持。

来源	目标	新增功能、增强或修复
全部	全部	从 JDK 8 升级到 Amazon Corretto JDK 11。有关更多信息，包括下载链接，请参阅《Amazon Corretto 11 用户指南》中的什么是 Amazon Corretto 11？
Teradata	Teradata	在 REGEXP_INSTR() 和 REGEXP_SUBSTR() 中实现了改进。
全部	全部	在评估报告中增加了有关用户数据库中可能存在的不一致之处的信息。
全部	SQL Server	SQL Server 2019：增加了对新索引属性 OPTIMIZE_FOR_SEQUENTIAL_KEY 的支持。 SQL Server 2017：增加了对图形数据库节点和边缘表类型的支持。 SQL Server 2016：增加了对 TEMPORAL TABLES 的支持。
Teradata	全部	增加了对 REGEXP_INSTR() 和 REGEXP_SUBSTR() 的支持。
全部	全部	实现了使用虚拟分区覆盖物理分区的功能。数据仓库提取器根据创建的虚拟分区提取数据。
Teradata	Amazon Redshift	使用“项目设置”、“另存为 SQL 和应用”、“下拉列表: 单个文件/多个文件”中的设置，实现了将源树的 SQL 按阶段保存到单个文件或多个文件中的能力。 在视图和过程转换中实现了改进。
Teradata	全部	增加了对 Teradata 16.20 版的支持

面向其他引擎的版本 1.0.640 更改

下表列出了面向其他源引擎的内部版本 1.0.640 更改。

来源	目标	新增功能、增强或修复
Sybase	RDS MariaDB 10.4	增加了对所有在线事务处理 (OLTP) 供应商的 RDS MariaDB 10.4 支持。
SAP ASE	MariaDB	实现了： • MariaDB 10.4 • EXECUTE IMMEDIATE 语句 • DEFAULT 定义 • CHECK 约束支持
SAP ASE	PostgreSQL 12.x	增加了对生成的列的支持。
全部	全部	从 JDK 8 升级到 Amazon Corretto JDK 11。有关更多信息，包括下载链接，请参阅 Amazon Corretto 11 用户指南中的什么是 Amazon Corretto 11？
全部	全部	在评估报告中增加了有关用户数据库中可能存在的不一致之处的信息。
SAP ASE	MySQL 8.0.17	增加了 CHECK 约束支持。
全部	SQL Server	SQL Server 2019：增加了对新索引属性 OPTIMIZE_FOR_SEQUENTIAL_KEY 的支持。 SQL Server 2017：增加了对图形数据库节点和边缘表类型的支持。 SQL Server 2016：增加了对 TEMPORAL TABLES 的支持。
Vertica	Amazon Redshift	增加了对分配方式 = AUTO 的支持。

来源	目标	新增功能、增强或修复
全部	全部	实现了使用虚拟分区覆盖物理分区的功能。数据仓库提取器根据创建的虚拟分区提取数据。
Amazon Redshift	Amazon Redshift	DML 语句中不受支持的内置函数将替换为 NULL 来作为占位符。
Sybase	PostgreSQL	增加了对本机函数的支持。
SAP ASE	MySQL/ Aurora MySQL	Aurora MySQL 的默认隔离级别是 REPEATABLE READ。无法在 Aurora MySQL 中设置 GLOBAL 隔离级别。只能更改会话范围。事务的默认行为是使用 REPEATABLE READ 和一致性读取。可能需要修改设计为通过 READ COMMITTED 运行的应用程序。或者，您可以明确地将默认值更改为 READ COMMITTED。
SAP ASE	PostgreSQL	增加了对不带扩展包的 CONVERT 函数（乐观）的支持。
SAP ASE	全部	增加了系统视图 sysindexes 模拟。 如果在没有指定 INTO 的过程中存在 SELECT 语句，则会为目标上的过程创建类型 refcursor 的参数 INOUT p_refcur。
Greenplum	Amazon Redshift	实现了 CREATE TEMPORARY TABLE，如下所示：

文档历史记录

下表介绍了 2018 年 1 月之后对 AWS Schema Conversion Tool（AWS SCT）用户指南的重要更改。

您可以订阅 RSS 源获取此文档更新的通知。

变更	说明	日期
AWS SCT 内部版本 #1.0.672	内部版本 1.0.672 支持将 Amazon RDS for PostgreSQL 15 作为目标，将 Microsoft SQL Server 版本 2022 作为源。它还在转换后的代码中增加了对新 Amazon Redshift 功能的支持，对 IBM Db2 for z/OS 源进行了多项转换改进，并解决了许多转换问题。	2023 年 5 月 8 日
AWS SCT 内部版本 #1.0.671	内部版本 1.0.671 支持从 Apache Oozie 迁移到 AWS Step Functions。它还增加了将 BigQuery 作为多服务器评估过程源的支持。此外，它还为作为源的 IBM Db2 for z/OS 添加了新的转换设置，并解决了许多转换问题。	2023 年 3 月 8 日
AWS SCT 内部版本 #1.0.670	内部版本 1.0.670 支持从 Hadoop 迁移到 Amazon EMR。它还增加了将 Azure Synapse Analytics 作为多服务器评估过程源的支持。此外，它还改进了嵌入在 Java 应用程序中的 SQL 代码的转换，并解决了许多转换问题。	2023 年 1 月 23 日

AWS SCT 内部版本 #1.0.669	内部版本 1.0.669 支持对从 Oracle 数据仓库迁移数据的本机分区。它还改进了多服务器评估过程，在数据提取代理中添加了新功能，并解决了许多转换问题。	2022 年 12 月 19 日
AWS SCT 内部版本 #1.0.668	内部版本 1.0.668 实现了自动虚拟分区，用于从 Greenplum 数据库迁移数据，并支持从 Snowflake 数据库到 Amazon Redshift 的数据迁移。它还改进了 C# 应用程序中嵌入的 SQL 代码的转换，并解决了许多转换问题。	2022 年 11 月 16 日
AWS SCT 内部版本 #1.0.667	内部版本 1.0.667 支持将 Informatica 提取、转换和加载（ETL）引擎作为迁移源。它还更新了扩展包版本，提高了 Amazon Redshift 支持的最低驱动程序版本，并解决了许多转换问题。	2022 年 10 月 13 日
AWS SCT 内部版本 #1.0.666	内部版本 1.0.666 通过添加对 MyBatis 框架的支持改进 Java 应用程序的转换。它还在扩展包中添加了新函数，增强了源元数据加载程序，并解决了许多转换问题。	2022 年 9 月 20 日

AWS SCT 内部版本 #1.0.665	内部版本 1.0.665 支持将 BigQuery 作为迁移源。它还实现了对新版本 Babelfish 功能配置文件的支持。此外，它还改善将数据仓库转换到 Amazon Redshift 的转换，并解决了许多转换问题。	2022 年 8 月 29 日
AWS SCT 内部版本 #1.0.664	内部版本 1.0.664 支持将 Amazon Redshift Serverless 作为迁移源或目标。它还在数据提取任务中实现了自动内存平衡，并修复了 AWS SCT 无法连接到 AWS Snowball 设备的错误。此外，它还增加了更改迁移规则中列排序规则的功能，改进了用户界面，并解决了许多转换问题。	2022 年 7 月 14 日
AWS SCT 内部版本 #1.0.663	内部版本 1.0.663 增加了对 适用于 Aurora PostgreSQL 的 Babelfish 的支持，并改进了多服务器评估报告功能。它还在迁移规则中添加了新功能，修复了两个加载程序错误，并解决了许多转换问题。	2022 年 6 月 20 日
AWS SCT 内部版本 #1.0.662	内部版本 1.0.662 在 C# 应用程序中实现 SQL 代码转换，并改进了多服务器评估报告工作流程。它还增加了多项转换改进，并解决了许多转换问题。	2022 年 5 月 19 日

AWS SCT 内部版本 #1.0.661	内部版本 1.0.661 支持将 IBM Db2 for z/OS 作为迁移源。它还增加了将提取、转换和加载（ETL）脚本转换为 AWS Glue Studio 的支持，并解决了多项转换问题。	2022 年 4 月 21 日
AWS SCT 内部版本 #1.0.660	内部版本 1.0.660 支持将 PostgreSQL 主要版本 14 和 MariaDB 10.6 作为迁移目标。它还增加了对实体化视图的 Oracle 索引转换的支持，并解决了许多转换问题。	2022 年 3 月 21 日
AWS SCT 内部版本 #1.0.659	内部版本 1.0.659 支持将 Aurora PostgreSQL 兼容版本上的 PostgreSQL 主要版本 13 作为迁移目标。它在 C# 应用程序中实现了 SQL 代码转换，增加了对 Oracle 统一审计的支持，并解决了许多转换问题。	2022 年 2 月 21 日
AWS SCT 内部版本 #1.0.658	内部版本 1.0.658 提供与 AWS Secrets Manager 的集成，并增加了对 Amazon Redshift 虚拟目标数据库平台的支持。它还增加了许多转换改进和错误修复。	2022 年 1 月 20 日
AWS SCT 内部版本 #1.0.657	内部版本 1.0.657 改善了从 Microsoft SQL Server 到 Aurora PostgreSQL 的兼容版本、Amazon RDS for PostgreSQL 和其他迁移目标的转换。它还增加了许多用户界面改进和错误修复。	2021 年 12 月 20 日

AWS SCT 内部版本 #1.0.656	内部版本 1.0.656 在一个项目中支持多个源数据库和目标数据库。它还增加了转换、优化策略、常规改进和一些错误修复。	2021 年 11 月 22 日
AWS SCT 内部版本 #1.0.655	内部版本 1.0.655 实现了将 Teradata FastExport 作业脚本转换为 Amazon Redshift RSQL，并将 Greenplum 支持的最低驱动程序版本提高到 42.2.5。它还增加了许多改进和错误修复。	2021 年 10 月 18 日
AWS SCT 内部版本 #1.0.654	内部版本 1.0.654 实现了 Shell、Teradata FastLoad 和 Teradata Basic Teradata Query（BTEQ）脚本到 Amazon Redshift RSQL 的转换。它还解决了许多转换问题，并增加了许多改进和错误修复。	2021 年 9 月 16 日
AWS SCT 内部版本 #1.0.653	内部版本 1.0.653 实现了在被调用的函数或过程中创建的动态 SQL 的转换。它还改进了加密例程的转换，并增加了许多改进和错误修复。	2021 年 8 月 10 日
AWS SCT 内部版本 #1.0.652	内部版本 1.0.652 在命令行界面中实现脚本命令模式并实施架构优化规则。它还增加了许多转换和性能改进以及错误修复。	2021 年 6 月 30 日

AWS SCT 内部版本 #1.0.651	内部版本 1.0.651 增加了多项改进和错误修复。它还提供对 AWS Schema Conversion Tool CLI 参考初始副本的访问权限。	2021 年 6 月 4 日
AWS SCT 内部版本 #1.0.650	内部版本 1.0.650 支持将 Amazon RDS for PostgreSQL 13 作为目标数据库，更新了提取代理。它还升级了 Microsoft SQL Server、Azure 和 Azure Synapse 支持的最低的 JDBC 驱动程序版本。此外，它还增加了许多转换改进和错误修复。	2021 年 4 月 30 日
AWS SCT 内部版本 #1.0.649	内部版本 1.0.649 支持将 MariaDB 10.5 作为目标数据库，并增强了 Oracle 内置函数的转换。它还增加了许多转换和性能改进以及错误修复。	2021 年 3 月 29 日
AWS SCT 内部版本 #1.0.648	内部版本 1.0.648 增加了许多转换改进和错误修复。	2021 年 2 月 22 日
AWS SCT 内部版本 #1.0.647	内部版本 1.0.647 增加了对 Amazon RDS 上数据库邮件功能的支持，实现了存储对象注释的加载和转换。它还添加了 AWS SCT 数据迁移服务评估器和 AWS SCT 向导，并实现了树筛选器用户界面。此外，它还在“评估报告”中添加了重新设计的部分，并添加了许多改进和错误修复。	2021 年 1 月 15 日

AWS SCT 内部版本 #1.0.646	内部版本 1.0.646 增加了对 INTERVAL 数据类型、标识列和光标转换的支持，并添加了许多改进和错误修复。	2020 年 12 月 28 日
AWS SCT 内部版本 #1.0.645	内部版本 1.0.645 支持 ETL SSIS 到 AWS Glue 的转换，并进行了许多改进和错误修复。	2020 年 11 月 16 日
AWS SCT 内部版本 #1.0.643-1.0.644	内部版本 1.0.644 增加了许多转换、性能和用户界面改进以及错误修复。	2020 年 10 月 14 日
AWS SCT 内部版本 #1.0.642	内部版本 1.0.642 实现了从 Microsoft SQL Server Integration Services 到 AWS Glue 的 ETL 包的转换，并增加了许多改进和错误修复。	2020 年 8 月 28 日
AWS SCT 内部版本 #1.0.641	为数据提取器添加了 SSL 支持。内部版本还包括许多改进和修复。	2020 年 7 月 17 日
AWS SCT 内部版本 #1.0.633-1.0.640	已从 JDK 8 升级到 Amazon Corretto JDK 11。增加了标识其他升级、更改和修复的表。	2020 年 6 月 22 日
AWS WQF 可用性	AWS SCT 不再提供 AWS Workload Qualification Framework（AWS WQF）工具以供下载。	2020 年 6 月 19 日

AWS SCT 内部版本 #1.0.632	SCT UI – 添加了新的选项卡，以显示应用脚本时发生的错误。从 SAP ASE 转换时，现在可以将源树另存为 SQL。改进了到 PostgreSQL 或 Aurora PostgreSQL 或 Redshift 的转换。	2019 年 11 月 19 日
AWS SCT 内部版本 #1.0.631 和 #1.0.630（组合）	在 Oracle 中更好地支持 ROWID，在 Microsoft SQL Server 和 SAP ASE 中更好地支持系统对象。更好地处理缺少的 SQL Server 模式指定符。更好地支持从 Greenplum 到 Redshift 的转换。改进了对移动到 Amazon Redshift、MariaDB、MySQL 和 PostgreSQL 时存储代码转换的支持。	2019 年 9 月 30 日
AWS SCT 内部版本 #1.0.629	支持存储过程从 Netezza 进行转换。改进了到 Amazon Redshift、DynamoDB、MySQL 和 PostgreSQL 的转换。增加了对 SAP ASE 12.5 作为源的支持。	2019 年 8 月 20 日
AWS SCT 内部版本 #1.0.628	支持对从 DB2、SQL Server 和 Oracle 的转换的服务模拟。转换到 Amazon Redshift 的增强功能，包括对游标和存储过程的更多支持。	2019 年 6 月 22 日
AWS SCT 内部版本 #1.0.627	支持从 SQL Server 转换到 Amazon Redshift 中的存储过程。转换为 PostgreSQL 11 和 MySQL 8.0 的增强功能。	2019 年 5 月 31 日

AWS SCT 内部版本 #1.0.626	现在支持将 PostgreSQL 11 和 MySQL 8.0 作为目标。现在支持 SAP ASE 15.5 作为源。	2019 年 4 月 26 日
AWS SCT 内部版本 #1.0.625	更新包括将 Teradata BTEQ 转换到 AWS Glue 的功能、对转换到支持 Oracle 兼容性模式的 MariaDB 10.3 的支持、对 SAP ASE 15.7 的支持以及模拟缺失功能的服务替代项。	2019 年 3 月 25 日
AWS SCT 内部版本 #1.0.624	更新包括将 Oracle ETL 转换到 AWS Glue 的功能以及对从 Microsoft SQL Server、Oracle 和 IBM Db2 LUW 到 Amazon RDS for MariaDB 的转换的支持。我们还增加了对从 SAP ASE 到 RDS for MySQL 和具有 MySQL 兼容性的 Amazon Aurora 的转换的支持。此外，我们还增加了在 Oracle Orafce 转换到 PostgreSQL 期间对 Orafce 扩展的支持。	2019 年 2 月 22 日
AWS SCT 内部版本 #1.0.623	更新包括转换 SAP ASE 数据库的功能以及将 T-SQL 脚本、DML 和 DDL 转换到等效代码或组件的功能。我们还添加了 Oracle 和 Microsoft SQL Server 模拟，以改进转换。	2019 年 1 月 25 日
AWS SCT 内部版本 #1.0.622	更新包括 Workload Qualification Framework，后者将分析整个迁移的工作负载，包括数据库和应用程序修改。	2018 年 12 月 20 日

AWS SCT 内部版本 #1.0.621	更新包括支持 Aurora PostgreSQL 10 作为目标，以及使用外部表选项从 Netezza 迁移的功能。	2018 年 11 月 21 日
AWS SCT 内部版本 #1.0.620	更新包括保存 SQL 脚本的功能，以及在迁移到 MySQL 时支持 Oracle 全局游标。	2018 年 10 月 22 日
AWS SCT 内部版本 #1.0.619	更新包括支持从 Apache Cassandra 迁移到 DynamoDB，并支持 Vertica 9 作为源。	2018 年 9 月 20 日
AWS SCT 内部版本 #1.0.618	更新包括扩展的评估报告，对转换 Oracle ROWID 的支持，以及对 SQL Server 用户定义的表的支持。	2018 年 8 月 24 日
AWS SCT 内部版本 #1.0.617	更新包括扩展的评估报告，对转换 Oracle ROWID 的支持，以及对 SQL Server 用户定义的表的支持。	2018 年 7 月 24 日
AWS SCT 内部版本 #1.0.616	更新包括在从 Oracle 转换至 Amazon RDS for Oracle、转换 Oracle 计划对象时对 RDS 的支持，以及对 Oracle 作业的支持。分区和 Db2 LUW 版本 10.1。	2018 年 6 月 26 日
AWS SCT 内部版本 #1.0.615	更新包含对 SQL Server 到 PostgreSQL GOTO 语句、PostgreSQL 10 分区以及 Db2 LUW 版本 10.1 的支持。	2018 年 5 月 24 日

AWS SCT 内部版本 #1.0.614	更新包括对 Oracle 到 Oracle 数据库链接、SQL Server 到 PostgreSQL 的内联函数以及模拟 Oracle 系统对象的支持。	2018 年 4 月 25 日
AWS SCT 内部版本 #1.0.613	更新包括对 Db2 LUW、SQL*Plus 文件转换以及 SQL Server Windows 身份验证的支持。	2018 年 3 月 28 日
AWS SCT 内部版本 #1.0.612	更新包括对自定义数据类型映射、Oracle 10 的架构比较以及 Oracle 到 PostgreSQL 的全局变量转换的支持。	2018 年 2 月 22 日
AWS SCT 内部版本 #1.0.611	更新包括对 Oracle 到 PostgreSQL 的动态语句、通过选择错误消息打开日志文件以及在树视图中隐藏架构这一功能的支持。	2018 年 1 月 23 日

早期更新

下表介绍了 2018 年 1 月前对 AWS Schema Conversion Tool（AWS SCT）用户指南的重要更改。

版本	更改	描述	更改日期
1.0.608	支持 Amazon S3 的 FIPS 端点	您现在可以请求 AWS SCT 通过使用符合联邦信息处理标准安全要求的 FIPS 端点连接到 Amazon S3 和 Amazon Redshift。有关更多信息，请参阅存储 AWS 凭证。	2017 年 11 月 17 日
1.0.607	支持 Amazon S3 的 FIPS 端点	您现在可以请求 AWS SCT 通过使用符合联邦信息处理标准安全要求的 FIPS 端点连接到 Amazon S3 和 Amazon Redshift。有关更多信息，请参阅存储 AWS 凭证。	2017 年 10 月 30 日

版本	更改	描述	更改日期
1.0.607	数据提取任务可以忽略 LOB	在创建数据提取任务时，您现在可以选择忽略大型对象 (LOB)，以减少提取的数据量。有关更多信息，请参阅创建、运行和监控 AWS SCT 数据提取任务。	2017 年 10 月 30 日
1.0.605	数据提取代理任务日志访问	您现在可以从 AWS Schema Conversion Tool 用户界面中的一个便捷链接访问数据提取代理任务日志。有关更多信息，请参阅创建、运行和监控 AWS SCT 数据提取任务。	2017 年 8 月 28 日
1.0.604	转换器增强功能	AWS Schema Conversion Tool 引擎经过增强，可为异构迁移提供改进的转换。	2017 年 6 月 24 日
1.0.603	数据提取代理支持筛选条件	您现在可以筛选提取代理从数据仓库中提取的数据。有关更多信息，请参阅在中创建数据迁移规则 AWS SCT。	2017 年 6 月 16 日
1.0.603	AWS SCT 支持其他数据仓库版本	您现在可以使用 AWS Schema Conversion Tool 将 Teradata 13 和 Oracle Data Warehouse 10 架构转换为等效的 Amazon Redshift 架构。有关更多信息，请参阅使用 AWS SCT 将数据仓库架构转换为 Amazon Redshift。	2017 年 6 月 16 日
1.0.602	数据提取代理支持其他数据仓库	您现在可以使用数据提取代理从 Microsoft SQL Server 数据仓库提取数据。有关更多信息，请参阅将本地数据仓库中的数据迁移到 Amazon Redshift。	2017 年 5 月 11 日
1.0.602	数据提取代理可以将数据复制到 Amazon Redshift	数据提取代理现在有三种上传模式。您现在可以指定是只提取数据，还是提取数据并只上传到 Amazon S3，还是提取、上传数据并直接复制到 Amazon Redshift。有关更多信息，请参阅创建、运行和监控 AWS SCT 数据提取任务。	2017 年 5 月 11 日

版本	更改	描述	更改日期
1.0.601	AWS SCT 支持其他数据仓库	您现在可以使用 AWS Schema Conversion Tool 将 Vertica 和 Microsoft SQL Server 架构转换为等效的 Amazon Redshift 架构。有关更多信息，请参阅使用 AWS SCT 将数据仓库架构转换为 Amazon Redshift。	2017 年 4 月 18 日
1.0.601	数据提取代理支持其他数据仓库	您现在可以使用数据提取代理从 Greenplum 、Netezza 和 Vertica 数据仓库提取数据。有关更多信息，请参阅将本地数据仓库中的数据迁移到 Amazon Redshift。	2017 年 4 月 18 日
1.0.601	数据提取代理支持其他操作系统	您现在可以在运行 macOS 和 Microsoft Windows 操作系统的计算机上安装数据提取代理。有关更多信息，请参阅安装提取代理。	2017 年 4 月 18 日
1.0.601	数据提取代理自动上传到 Amazon S3	数据提取代理现在自动将提取的数据上传到 Amazon S3。有关更多信息，请参阅数据提取任务输出。	2017 年 4 月 18 日
1.0.600	数据提取代理	您现在可以安装数据提取代理，以便从数据仓库提取数据并将其准备用于 Amazon Redshift。您可以使用 AWS Schema Conversion Tool 注册代理并为其创建数据提取任务。有关更多信息，请参阅将本地数据仓库中的数据迁移到 Amazon Redshift。	2017 年 2 月 16 日
1.0.600	客户反馈	您现在可以提供关于 AWS Schema Conversion Tool 的反馈。您可以提交错误报告、提交功能请求，也可以提供常规信息。有关更多信息，请参阅提供反馈。	2017 年 2 月 16 日

版本	更改	描述	更改日期
1.0.502	与 AWS DMS 集成	您现在可以使用 AWS Schema Conversion Tool 创建 AWS DMS 终端节点和任务。您可以从 AWS SCT 中运行和监控任务。有关更多信息，请参阅配合使用 AWS SCT 和 AWS DMS。	2016 年 12 月 20 日
1.0.502	作为目标数据库的兼容 PostgreSQL 的 Amazon Aurora	AWS Schema Conversion Tool 现在支持作为目标数据库的兼容 PostgreSQL 的 Amazon Aurora。有关更多信息，请参阅使用 AWS SCT 转换数据库架构。	2016 年 12 月 20 日
1.0.502	支持配置文件	您现在可以将不同的配置文件存储在 AWS Schema Conversion Tool 中并在它们之间轻松切换。有关更多信息，请参阅将 AWS 服务配置文件存储在 AWS SCT 中。	2016 年 12 月 20 日
1.0.501	支持 Greenplum 数据库和 Netezza	您现在可以使用 AWS Schema Conversion Tool 将数据仓库架构从 Greenplum 数据库和 Netezza 转换为 Amazon Redshift。有关更多信息，请参阅使用 AWS SCT 将数据仓库架构转换为 Amazon Redshift。	2016 年 11 月 17 日
1.0.501	Redshift 优化	您现在可以使用 AWS Schema Conversion Tool 优化 Amazon Redshift 数据库。有关更多信息，请参阅使用 AWS SCT 优化 Amazon Redshift。	2016 年 11 月 17 日
1.0.500	映射规则	使用 AWS Schema Conversion Tool 转换架构之前，您现在可以设置以下操作的规则：更改列数据类型、将对象从一个架构复制到另一架构，以及更改对象名称。有关更多信息，请参阅在 AWS SCT 中创建迁移规则。	2016 年 10 月 4 日

版本	更改	描述	更改日期
1.0.500	迁移到云	您现在可以使用 AWS Schema Conversion Tool 将现有的本地数据库架构复制到运行相同引擎的 Amazon RDS 数据库实例。您可以使用此功能来分析迁移到云和更改许可证类型的潜在成本节省。有关更多信息，请参阅使用 AWS SCT 创建评估报告。	2016 年 10 月 4 日
1.0.400	数据仓库架构转换	您现在可以使用 AWS Schema Conversion Tool 将数据仓库架构从 Oracle 和 Teradata 转换为 Amazon Redshift。有关更多信息，请参阅使用 AWS SCT 将数据仓库架构转换为 Amazon Redshift。	2016 年 7 月 13 日
1.0.400	应用程序 SQL 转换	您现在可以使用 AWS Schema Conversion Tool 通过 C++、C#、Java 或其他应用程序代码转换 SQL。有关更多信息，请参阅使用 AWS SCT 转换应用程序 SQL。	2016 年 7 月 13 日
1.0.400	新特征	AWS Schema Conversion Tool 现在包含一个扩展包和一个向导，可帮助您安装、创建和配置 AWS Lambda 函数和 Python 库，以提供电子邮件、作业调度和其他功能。有关更多信息，请参阅 使用 AWS SCT 扩展包中的 AWS Lambda 函数 和 为 AWS SCT 扩展包使用自定义库：	2016 年 7 月 13 日
1.0.301	SSL 支持	使用 AWS Schema Conversion Tool 时，您现在可以使用安全套接字层 (SSL) 连接到您的源数据库。	2016 年 5 月 19 日
1.0.203	新特征	添加了对作为源数据库的 MySQL 和 PostgreSQL 的支持，以便进行转换。	2016 年 4 月 11 日

版本	更改	描述	更改日期
1.0.202	维护版本	添加了支持，以便编辑为目标数据库引擎生成的转换后的 SQL。在源数据库和目标数据库实例树视图中添加了改进的选择功能。添加了支持，以便使用透明网络底层 (TNS) 名称连接到 Oracle 源数据库。	2016 年 3 月 2 日
1.0.200	维护版本	添加了对作为目标数据库引擎的 PostgreSQL 的支持。添加了一种功能，可将转换后的架构生成为脚本并可将脚本保存到文件中，然后再将该架构应用于目标数据库实例。	2016 年 1 月 14 日
1.0.103	维护版本	添加了离线项目功能、检查新版本的功能以及内存和性能管理。	2015 年 12 月 2 日
1.0.101	维护版本	添加了 Create New Database Migration Project 向导。添加了将数据库迁移评估报告保存为 PDF 文件的功能。	2015 年 10 月 19 日
1.0.100	预览版	为 AWS Schema Conversion Tool 预览版提供了用户指南。	2015 年 10 月 7 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。